INVESTIGATION OF COMBUSTIVE FLOWS AND DYNAMIC MESHING IN COMPUTATIONAL FLUID DYNAMICS

A Thesis

by

STEVEN B. CHAMBERS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2004

Major Subject: Aerospace Engineering

INVESTIGATION OF COMBUSTIVE FLOWS AND DYNAMIC MESHING IN COMPUTATIONAL FLUID DYNAMICS

A Thesis

by

STEVEN B. CHAMBERS

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Paul G. A. Cizmas (Chair of Committee)	Leland A. Carlson (Member)
Raytcho D. Lazarov (Member)	Othon K. Rediniotis (Member)
Walter E. Haisler (Head of Department)	

December 2004

Major Subject: Aerospace Engineering

ABSTRACT

Investigation of Combustive Flows and
Dynamic Meshing in Computational Fluid Dynamics. (December 2004)
Steven B. Chambers, B.S., Texas A&M University
Chair of Advisory Committee: Dr. Paul G. A. Cizmas

Computational Fluid Dynamics (CFD) is a field that is constantly advancing. Its advances in terms of capabilities are a result of new theories, faster computers, and new numerical methods. In this thesis, advances in the computational fluid dynamic modeling of moving bodies and combustive flows are investigated. Thus, the basic theory behind CFD is being extended to solve a new class of problems that are generally more complex. The first chapter that investigates some of the results, chapter IV, discusses a technique developed to model unsteady aerodynamics with moving boundaries such as flapping winged flight. This will include mesh deformation and fluid dynamics theory needed to solve such a complex system. Chapter V will examine the numerical modeling of a combustive flow. A three dimensional single vane burner combustion chamber is numerically modeled. Species balance equations along with rates of reactions are introduced when modeling combustive flows and these expressions are discussed. A reaction mechanism is validated for use with *in situ* reheat simulations. Chapter VI compares numerical results with a laminar methane flame experiment to further investigate the capabilities of CFD to simulate a combustive flow. A new method of examining a combustive flow is introduced by looking at the solutions ability to satisfy the second law of thermodynamics. All laminar flame simulations are found to be in violation of the entropy inequality.

To Greg and Wendy

ACKNOWLEDGMENTS

No proper acknowledgment can be written without thanking my adviser, Dr. Paul Cizmas. Every step of the way he has been a true mentor and a friend. Long after this work is forgotten, I will still remember the person he is. I also would like to thank the members of my committee: Leland Carlson, Raytcho Lazarov and Othon Rediniotis. They challenged me in ways I had never known or wanted and made me all the better because of it. Additionally, I would like to thank Dr. John Slattery. The many discussions with him assisted in my understanding of the material. I would like to thank my peers: Roshawn Bowers, Joaquin Gargoloff, Jason Guarnieri, Kyu-sup Kim, Aditya Murthi, Josh O'Neil, Celerino Resendiz, Amarnath Sambasivam, Leslie Weitz, and Tao Yuan. Lastly, I would like to thank my family. Without their love and support, none of this would be possible.

NOMENCLATURE

2D	–	Two-dimensional
3D	–	Three-dimensional
a	–	Summation of the entropy inequality expression over all the cells in violation of the second law
A_r	–	Pre-exponential factor
c	–	Total molar density
$C_{j,r}$	–	Molar concentration of species j in reaction r
cm	–	Centimeters
$c_{p,j}$	–	Constant pressure specific heat of species j
CFD	–	Computational fluid dynamics
D	–	Diameter of circular cylinder
$\overline{\overline{D}}$	–	Rate of deformation tensor
$\vec{d_i}$	–	Intermediate term defined on (p.450) of [Slattery]
$\mathcal{D}_{ij}$	–	Binary diffusion coefficient
D_{ij}	–	Matrix of binary diffusion coefficients
$D_{i,m}$	–	Diffusion coefficient for species i in mixture
$D_{T,i}$	–	Thermal diffusion coefficient
E_r	–	Activation energy for reaction
$F(\phi)$	–	Spatial discretized function
$\vec{F_i}$	–	Force on mesh node i

$\vec{g}$ – Gravitational acceleration

h – Enthalpy

h_j^0 – Enthalpy of formation of species j

I – Turbulence intensity

$\overline{\overline{I}}$ – Identity matrix

in – Inches

$\vec{J_i}$ – Mass diffusion flux for species i

k – Turbulent kinetic energy, thermal conductivity

k_B – Boltzmann's constant

k_{eff} – Effective heat conductivity

k_{ij} – Spring constant between nodes i and j

$k_{f,r}$ – Forward rate constant for reaction r

$k_{b,r}$ – Backward rate constant for reaction r

K_r – Reaction equilibrium constant for reaction r

L – Hydraulic diameter

m – Meters

mm – Millimeters

M_i – Molecular mass of species i

N – Number of chemical species present in the system

n_i – Number of neighboring nodes connected to node i

N_r – Number of chemical species in reaction r

N_s	–	Total number of chemical species in one chemical reaction
P	–	Pressure
Pa	–	Pascals
P_R	–	Reduced pressure
q	–	Total number of cells that violate the second law of thermodynamics
$r_{i,r}$	–	Mass rate of production of species i by chemical reaction r
R	–	Universal gas constant
$\hat{R}_{i,r}$	–	Arrhenius molar rate of production of species i in reaction r
R_i	–	Species mass rate of production by all chemical reactions
Re	–	Reynolds number
Sc_t	–	Turbulent Schmidt number
S_h	–	Heat energy due to chemical reaction
S_i	–	Arbitrary specification of chemical species i, source term of component i in momentum equation
St	–	Strouhal number
T	–	Temperature
T^*	–	Dimensionless temperature
T_{ref}	–	Reference temperature
UDF	–	User defined function
U_e	–	Boundary layer edge velocity

U_∞ – Freestream velocity

u_i – Velocity vector using index notation

$\overline{U}$ – Mean flow velocity

$u^{'}$ – Root mean square of velocity fluctuations

$\vec{v}$ – Velocity vector

V – Cell volume

w – Calculation of entropy inequality at a single cell

X_i – Mole fraction for species i

Y_i – Mass fraction for species i

α_δ – Cell height factor

β_r – Temperature exponent

δ – Cell height

δ_{ideal} – Ideal cell height

ΔG° – Gibbs energy change

ΔH° – Standard enthalpy change of reaction

Δt – Time step

$\Delta \vec{x}_j$ – Displacement of node j

ϵ – Characteristic energy

$\vec{\epsilon}$ – Intermediate term defined on (p.449) of [Slattery]

$\eta^{'}_{j,r}$ – Forward rate exponent of species j in reaction r

$\eta''_{j,r}$ – Backward rate exponent of species j in reaction r

γ_B – Activity coefficient

μ – Viscosity

μ_i – Viscosity of species i

μ_t – Turbulent viscosity

$\nu'_{i,r}$ – Stoichiometric coefficient for reactant i in reaction r

$\nu''_{i,r}$ – Stoichiometric coefficient for product i in reaction r

Ω_D – Collision integral

ϕ – Arbitrary scalar quantity

ϕ_f – Values of ϕ convected through face f

ρ – Density

σ_i – Collision diameter

$\bar{\bar{\tau}}$ – Viscous stress tensor used within FLUENT

$\overline{\overline{T}}$ – General expression for stress tensor for a Newtonian fluid

TABLE OF CONTENTS

CHAPTER Page

I INTRODUCTION 1

A. CFD of Combustive Flows for Turbomachinery Applications 1
B. Dynamic Mesh Modeling and Aeroelastic Considerations . 2

II PHYSICAL MODELS OF FLUID MECHANICS AND COMBUSTION 4

A. Description of Fundamental Fluid Flow Equations 4
1. Continuity Equation 4
2. Navier-Stokes Equations 5
3. Energy Equation 7
B. Transport Equations Used for Combustion 9
1. Species Transport Model 9
2. Multicomponent Species Transport 10
3. Reaction Rate Expression 12

III NUMERICAL MODEL 16

A. Description of Solution Method 16
1. Linearization 17
2. Discretization 18
a. Second-Order Upwinding Scheme 19
b. Time Discretization 19
3. Pressure-Velocity Correction 20
a. PISO Pressure-Correction Scheme 21
4. Moving Deforming Grid 22
a. Spring-Based Smoothing Method 23
b. Local Remeshing Method 24

IV RESULTS FOR MOVING DEFORMING MESH 27

A. Grid Generation for Moving Deforming Mesh 27
B. User Defined Functions 29
C. Low Speed Unsteady FLUENT Solution Investigation . . . 32
D. Application of Moving Deforming Mesh to Flapping Flight 42
E. Conclusions and Future Applications 49

CHAPTER Page

V VALIDATION OF COMBUSTION MODEL FOR *IN SITU* REHEAT WITH 3-D METHANE INJECTION VANE 51

A. Experimental Setup 52
B. Numerical Boundary Conditions 54
C. Grid Generation 58
D. Combustion Model Used in Simulation 62
1. Chemical Model A 62
2. Chemical Model B 63
E. Results 64
1. Results of 3D Injector Simulation with Chemical Model A 65
2. Results of 3D Injector with Chemical Model B 72
F. Conclusions and Recommendations 76

VI NUMERICAL INVESTIGATION OF A LAMINAR FLAME . . 82

A. Experimental Setup 83
B. Combustion Model 87
C. Numerical Model 88
1. Boundary Conditions 89
2. Description of Computational Grid 91
3. Temperature Limiter 97
D. Entropy Inequality Expression with Numerical Implementation 98
1. First Term 98
2. Second Term 99
3. Third Term 105
4. Fourth Term 110
E. Results 111
1. Comparison with Experimental Results 111
2. Entropy Inequality Results 118
a. First Term 119
b. Second Term 120
c. Third Term 121
d. Fourth Term 124
e. Overall Inequality Satisfaction 126
F. Conclusions and Future Work 132

VII CONCLUSIONS 134

CHAPTER Page

A. Discussion of Physical Model and Numerical Implementation 134
B. Investigation into Moving Rigid Boundaries in CFD 135
C. Validation of Simple Combustion Model for *In Situ* Reheat Investigation 136
D. Further Investigation of Simple Combustion Model and Its Ability to Satisfy the Second Law of Thermodynamics . 137
E. Further Applications 138

REFERENCES 141

APPENDIX A 145

VITA 163

LIST OF TABLES

TABLE		Page
I	Flow parameters used in simulation	37
II	Flow parameters used for flapping arc simulation	46
III	Experimental inlet gas mixture molar composition percentage	55
IV	Experimental fuel injection mixture molar composition percentage	55
V	Input data for vane-burner	58
VI	Numerical grid size information	60
VII	Species mole fraction % at 0.311 m downstream using chemical model A	66
VIII	Species mole fraction % at 0.654 m downstream using chemical model A	66
IX	Temperature values at axial locations using chemical model A. Experimental value at 0.836 m is 1478K	67
X	Species mole fraction % at 0.311 m downstream using chemical model B	74
XI	Species mole fraction % at 0.654 m downstream using chemical model B	74
XII	Temperature values at axial locations using chemical model B. Experimental value at 0.836 m is 1478K	75
XIII	Sandia velocity inlet boundary conditions specification	90
XIV	Sandia pressure outlet boundary conditions specification	91

LIST OF FIGURES

FIGURE		Page
1	Sample of moving deforming mesh. Picture at far left is the initial grid, middle picture shows the cylinder when it has reached its peak displacement upward and the far right picture shows the cylinder at the bottom of its translation.	29
2	Sample of a user defined function (UDF) that defines a vertical sinusoidal movement to a cylinder.	33
3	Outer domain of circular cylinder mesh.	34
4	Boundary layer mesh surrounded by unstructured grid.	35
5	Non-dimensional lift versus time for 5 *sec* time steps.	39
6	Non-dimensional lift versus time for 10 *sec* time steps.	39
7	Non-dimensional lift versus time for 10 *sec* time steps with 10 sub-iterations.	40
8	Vorticity magnitude and grid resolution.	41
9	Moving portion of arc grid.	43
10	Translational and angular velocity of forward flying hornet.	44
11	Mesh plots showing grid resolution during flapping motion.	45
12	Velocity magnitudes at different instances in the cycle of the flapping motion.	47
13	Static pressure contour of entire domain showing pressure build-up at exit.	48
14	Experimental setup for single-vane burner.	52
15	Idealized experimental apparatus.	53

FIGURE Page

16 Combustion probe geometry. 54

17 Idealized illustration of numerical domain. 59

18 Detail of fuel injector. 61

19 Series of temperature contour plots using combustion model A. . . . 69

20 Series of methane mass fraction contour plots using combustion model A. 70

21 Series of carbon monoxide mass fraction contour plots using combustion model A. 71

22 Series of temperature contour plots with reversible reaction defined. . 77

23 Series of methane mass fraction contour plots with reversible reaction defined. 78

24 Series of carbon monoxide mass fraction contour plots with reversible reaction defined. 79

25 Temperature contour plot of turbine *in situ* reheat simulation. 80

26 Experimental setup of Sandia combustion flame facility. 84

27 Detail setup of laminar combustion flame. 85

28 Photograph of actual laminar flame. 86

29 Idealized pictorial description of initial laminar flame grid. 93

30 Temperature contour plot with boundary located at the exit of the tube. 94

31 Idealized illustration of final numerical domain of laminar flame simulation. 95

32 Temperature contour plot for each temperature limiter. From left to right, solutions are shown for temperature limits of 2025, 2300, 2600, and 2900K. 112

FIGURE Page

33 Comparison with experimental data at 25 *mm*. 114

34 Comparison with experimental data at 50 *mm*. 116

35 Comparison with experimental data at 100 *mm*. 117

36 Contour plot showing values of first term of entropy inequality. . . . 119

37 Contour plot showing values of second term of entropy inequality. . . 121

38 Contour plot showing values of the third term of entropy inequality. . 122

39 Contour plot showing locations within the domain where the third term is greater than zero. 123

40 Contour plot showing temperature variation of Sandia simulation. . . 125

41 Contour plot showing values of fourth term of entropy inequality. . . 125

42 Comparison of the advancement of the solution and the number of points which violated the second law. 127

43 Comparison of the advancement of the solution and the magnitude of the entropy violation. 128

44 Locations where entropy inequality is not satisfied for temperature limiter of $2025K$ at 15000 iterations. 130

45 Locations where entropy inequality is not satisfied for temperature limiter of $2300K$ at 15000 iterations. 130

46 Locations where entropy inequality is not satisfied for temperature limiter of $2600K$ at 15000 iterations. 131

47 Locations where entropy inequality is not satisfied for temperature limiter of $2900K$ at 1000 iterations. 131

CHAPTER I

INTRODUCTION

A. CFD of Combustive Flows for Turbomachinery Applications

Innovations in computational technologies have opened the door for advances in the area of power generation by way of computational fluid dynamics (CFD). Traditional research on turbomachinery has involved fabrication and testing of actual systems, which is often expensive and time consuming. CFD allows designers to increase efficiency and decrease pollution levels of turbomachinery systems without the expense of fabricating test articles. CFD is a technique used to perform aerodynamic research, which is used to enhance engine efficiency by improving the airflow through the engine. Perhaps the most common method of performing computational fluid dynamics is discretizing the physical domain, whether it is a compressor or turbine, and the application of numerical simulation of the fluid flow through the system using the Navier-Stokes equations. CFD research has lead to the development of new airfoil shapes for turbine and compressor blades and stators which increase the overall efficiency of the turbomachinery system.

While CFD has been used in the past to calculate the air through a turbine to increase efficiency, it is the objective of this work to use CFD to help develop an improved way of calculating combustion within a turbine. In an attempt to increase the thrust-to-weight ratio and decrease the thrust specific fuel consumption, turbomachinery designers are facing the fact that the combustor residence time can become shorter than the time required to complete combustion. Thus, the combustion process could continue into the turbine, a process which is often considered

---

The journal model is *Journal of Propulsion and Power.*

undesirable. However, a thermodynamic cycle analysis performed demonstrates the benefits of extending the combustion into the turbine in order to increase the specific power and thermal efficiency.[1] The process of combustion in the turbine is called *in situ* reheat. In order to accurately capture the combustion phenomena an accurate numerical model for combustion must be used. Developing an accurate yet cost effective combustion model that will be used to numerically investigate the feasibility of *in situ* reheat is the focus of this research.

B. Dynamic Mesh Modeling and Aeroelastic Considerations

Aeroelastic considerations in aircraft systems is a rapidly growing topic. Aeroelasticity is often defined as a science which studies the mutual interaction between aerodynamic forces and elastic forces, and the influence of this interaction on airplane design.[2] However, aeroelasticity is not only limited to aircraft. The most famous example of the importance of aeroelasticity is the Tacoma Narrows bridge. Because no thought was given to how the bridge would interact with its environment, the bridge had a catastrophic failure in November 7, 1940. The bridge collapsed because wind induced vibrations were not taken into account during its structural design. This was essentially the birth of aeroelasticity. But with the development of ever faster and larger computing power, aeroelasticity is being included into designs now more than ever. Its inclusion into the aircraft design of the future is essential so that possible failures in the aircraft are known before they take to the air.

Aeroelastic calculations have three main portions. The first step is calculating the flow behavior around an object. The flow behavior is then transfered to the structure in terms of aerodynamic loads acting on the structure. The second step is the transfer of the aerodynamic loads to the structural model. Once the loads

are fed into a structural model the last step is to calculate the displacements. The displacements are the shifting of the structure due to the aerodynamic loads currently acting on it. These displacements are then fed back to the aerodynamic solver to find new aerodynamic loads. This process is repeated as long as the aerodynamic loads are changing. The method in which the forces and displacements are transferred from flow solver to structural solver is just as important as the flow solution and structural solution themselves. Because of this fluid-structure interaction, it becomes necessary to have a dynamic mesh model which can be used to model flows where the shape of the domain is changing with time due to motion on the domain boundaries. Dynamic mesh modeling is the portion of the aeroelastic problem that has been investigated in this research. Accurate dynamic mesh modeling will provide the basis for the numerical modeling of highly deforming aircraft and eventually even flapping winged aircraft.

CHAPTER II

PHYSICAL MODELS OF FLUID MECHANICS AND COMBUSTION

CFD is a numerical tool used to describe the motion of a fluid flow. Before any computation is performed, it is necessary to develop the theory behind what the computer is asked to compute. This chapter will provide the physical theory that is necessary to numerically compute combustive flows. It will begin by discussing the governing equations of fluid mechanics and will end with a discussion of the added equations which are used to simulate a combustion flow.

A. Description of Fundamental Fluid Flow Equations

In this section the fundamental fluid flow transport equations are discussed. These equations include the continuity equation, Navier-Stokes equations, and when appropriate, the viscous flow energy equation. The introduction here will only be a brief layout of what is often used in a fluid dynamics solver, and more specifically what is used in FLUENT. All computations are performed with this commercially available fluid dynamics software. A more general introduction to these equations is found in [Tannhill, Anderson, Pletcher].[3] A description of the fundamental transport equations will be introduced in this chapter, while the next chapter will outline the numerical method used to solve the governing equations.

1. Continuity Equation

The continuity equation, or conservation of mass, for a compressible fluid in a control volume is given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0. \tag{2.1}$$

Here ρ is the fluid density and $\vec{v}$ is the velocity vector. This expression of the conservation of mass allows for variations in time and is written in partial differential form. Therefore it is valid at every point inside the flow domain.

2. Navier-Stokes Equations

The Navier-Stokes equations may be written as

$$\frac{\partial}{\partial t}(\rho\vec{v}) + \nabla \cdot (\rho\vec{v}\vec{v}) = -\nabla P + \rho\vec{g} + \nabla \cdot \bar{\bar{\tau}}, \tag{2.2}$$

where $\vec{g}$ is the gravitational acceleration, P is the flow pressure and $\overline{\overline{\tau}}$ is the stress tensor defined by

$$\overline{\overline{\tau}} = \mu\Big[(\nabla\vec{v} + \nabla\vec{v}^{\top}) - \frac{2}{3}(\nabla \cdot \vec{v})\overline{\overline{I}}\Big]. \tag{2.3}$$

μ is the dynamic viscosity and $\overline{\overline{I}}$ is the identity matrix. This representation of the stress tensor makes the approximation of the bulk viscosity being equal to 2/3 the dynamic viscosity. For simulations using a moving and deforming mesh the dynamic viscosity will be held constant. For problems investigating moving and deforming meshes, temperature changes are not the focus of the research, and thus a constant dynamic viscosity is a reasonable assumption for a laminar flow where the only heating effects are due to viscosity. For combustion calculations a constant dynamic viscosity is no longer an ideal assumption and thus it will be calculated from kinetic theory.

The dynamic viscosity for a specific chemical species is given by the following expression,[4]

$$\mu_i = 2.67\text{x}10^{-6}\frac{\sqrt{M_i T}}{\sigma_i^2 \Omega_{D_i}}. \tag{2.4}$$

This is the Chapman-Enskog viscosity equation and the subscript i stands for a

particular species. M_i is the molecular mass of the species being considered, σ_i is the collision diameter and is given in units of Angstroms. Ω_{D_i} is obtained as a complex function of a dimensionless temperature, T^*. At this point the subscript i is dropped because all of the following definitions are valid for individual species. The expression for Ω_D is an empirical equation given as

$$\Omega_D = [A(T^*)^{-B}] + C[e^{-DT^*}] + E[e^{-FT^*}]. \tag{2.5}$$

T^* is defined as

$$T^* = \frac{T}{(\epsilon/k_B)}, \tag{2.6}$$

and A = 1.16145, B = 0.14874, C = 0.52487, D = 0.77320, E = 2.16178 and F = 2.43787. Equation (2.5) is valid from $0.3 \leq T^* \leq 100$.[5]

ϵ/k_B is the characteristic energy divided by Boltzmann's constant and is one of a group of parameters called the Leonard Jones parameters. ϵ/k_B and σ are listed in [Reid Prausnitz Poling] for many different species.[5]

At this point, the dynamic viscosity has only been introduced for each species. In order to define μ in (2.2), the dynamic viscosity for the mixture must be calculated from the dynamic viscosity of each species found in the mixture. This is done with the help of an ideal-gas mixing-law. The dynamic viscosity for a mixture is given by

$$\mu = \sum_i \frac{X_i \mu_i}{\sum_j X_j \phi_{ij}}, \tag{2.7}$$

where X_i is the mole fraction of species i.[4] Here the mole fraction is defined as the number of moles of a local constituent divided by the total number of moles of all local constituents in the mixture.[6] ϕ_{ij} is an intermediate quantity and is defined as a matter of convenience by

$$\phi_{ij} = \frac{\left[1 + \left(\frac{\mu_i}{\mu_j}\right)^{\frac{1}{2}} \left(\frac{M_j}{M_i}\right)^{\frac{1}{4}}\right]^2}{\left[8\left(1 + \frac{M_i}{M_j}\right)\right]^{\frac{1}{2}}}. \tag{2.8}$$

3. Energy Equation

As shown in the viscosity calculation, when temperature effects are important, such as with the combustion analysis, the energy equation must be added to the governing equations. The conservation of energy equation is shown in the following form:

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\vec{v}(\rho E + P)) = \nabla \cdot \left(k_{\text{eff}} \nabla T - \sum_j h_j \vec{J}_j + (\overline{\overline{\tau}}_{\text{eff}} \cdot \vec{v}) \right) + S_h, \tag{2.9}$$

where k_{eff} is the effective heat conductivity, which, when appropriate, is composed of a turbulent and laminar component. $\vec{J}_i$ is the diffusion mass flux vector of the species i and is discussed in more detail when the multicomponent species model is introduced later in this chapter. The first three terms on the right hand side of (2.9) represent energy due to conduction, species diffusion, and viscous dissipation, respectively.[4] S_h is a source term that takes into account the heat released or consumed by a chemical reaction. This term is added anytime combustion is simulated.

E is the total energy and has the following expression,

$$E = h - \frac{p}{\rho} + \frac{v^2}{2}, \tag{2.10}$$

where h is the enthalpy, which for an ideal gas in a multicomponent flow is calculated as

$$h = \sum_j Y_j h_j, \tag{2.11}$$

Y_j is the species mass fraction and h_j is defined as

$$h_j = \int_{T_{ref}}^{T} c_{p,j}\, dT. \tag{2.12}$$

T_{ref} is the reference temperature which is usually chosen to be $298.15K$. The specific heat is computed using a piecewise-polynomial expression that is dependent on temperature. Therefore the expression for specific heat resembles:

$$c_p(T) = A_1 + A_2T + A_3T^2 + A_4T^3 + A_5T^4 \quad (\text{for} \quad \mathrm{T_{min}} < \mathrm{T} < \mathrm{T_{max}}), \tag{2.13}$$

for a given temperature range. Another set of coefficients is needed for the next temperature range. The coefficients are available for each species that are found in the domain. Default coefficient values found in FLUENT were used, and checked using [McBride Gordon & Reno].[7] Each species had a polynomial expression for specific heat for the temperature range of $300K$ to $1000K$ and then another expression from $1000K$ to $5000K$.

The source term in (2.9) has the expression:

$$S_h = -\sum_j \Big(\frac{h_j^0}{M_j} + \int_{T_{\mathrm{ref,j}}}^{T} c_{p,j}\, dT\Big) R_j. \tag{2.14}$$

h_j^0 is the enthalpy of formation of species j and R_j is the net rate of production of species j due to all chemical reactions.[4] Further information about R_j will be given when the reaction rate expression is introduced.

A brief overview of the governing equations of fluid dynamics has been given. Some of the specific terms in the equations which are critical to the current research have been explained in more detail. The terms not explicitly discussed can be found in the FLUENT Users Guide.[4] The next section will introduce the combustive model which was used in the presented work.

B. Transport Equations Used for Combustion

One method used to calculate a combustive flow is to include both the Navier-Stokes equations and the species conservation equations in a numerical simulation. In addition to species conservation equations there must be a mathematical way to represent the reaction rates of different chemical reactions. The expressions which have historically been used to determine reaction rates are empirical algebraic models obtained through experimental testing. A description of these empirical models will be given.

1. Species Transport Model

The conservation equation for chemical species can be written as such,

$$\frac{\partial}{\partial t}(\rho Y_i) + \nabla \cdot (\rho \vec{v} Y_i) = -\nabla \cdot \vec{J_i} + R_i. \tag{2.15}$$

ρ is the local density and Y_i is the local mass fraction of each species. The local mass fraction is defined as the mass of a local constituent divided by the total mass of all local constituents in the mixture.[6] Thus, the mass fraction changes at different cell locations within the domain. A consequence of the conservation of mass is that at a point, or discrete cell, the mass fractions of all the species present must sum to unity. Therefore, equation (2.15) is only solved for $N-1$ species. The last species, or the Nth species, is calculated after all of the other species by requiring the sum of the mass fractions at a point to be equal to one.

As mentioned earlier, $\vec{J_i}$ is the mass diffusion flux of species i. A careful treatment of mass diffusion flux vector in the species transport and energy equations is important in diffusion-dominated laminar flows. FLUENT has the ability to model full multicomponent species transport and this method is used to model laminar-flow diffusion. The next section will discuss some of the details that are used when full

multicomponent diffusion is used.

2. Multicomponent Species Transport

FLUENT uses the Maxwell-Stefan equations to obtain the expression for the diffusive mass flux.[4] When a dilute gas is assumed, the Maxwell diffusion coefficients are interpreted as binary diffusion coefficients.[8] With the help of kinetic theory, binary diffusion coefficients are much easier to calculate than the Maxwell diffusion coefficients. The formulation of the binary diffusion coefficients will be given in chapter VI, as the they are necessary to calculate the diffusive mass flux vector. The diffusive mass flux vector, $\vec{J_i}$, may be written as,[9]

$$\vec{J_i} = -\sum_{j=1}^{N-1} \rho D_{ij} \nabla Y_j - D_{T,i} \frac{\nabla T}{T}. \tag{2.16}$$

where Y_j is the mass fraction of species j. D_{ij} is defined as,

$$D_{ij} = [D] = [A]^{-1}[B], \tag{2.17}$$

where the $[A]$ and $[B]$ matrices are defined in equations (2.18)-(2.20).

$$A_{ii} = -\left(\frac{X_i}{\mathcal{D}_{iN}} \frac{M_{mix}}{M_N} + \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{X_j}{\mathcal{D}_{ij}} \frac{M_{mix}}{M_i} \right) \tag{2.18}$$

$$A_{ij} = X_i \left(\frac{1}{\mathcal{D}_{ij}} \frac{M_{mix}}{M_j} - \frac{1}{\mathcal{D}_{iN}} \frac{M_{mix}}{M_N} \right) \tag{2.19}$$

$$B_{ii} = -\left(X_i \frac{M_{mix}}{M_N} + (1 - X_i) \frac{M_{mix}}{M_i} \right) \tag{2.20}$$

$$B_{ij} = X_i \left(\frac{M_{mix}}{M_j} - \frac{M_{mix}}{M_N} \right) \tag{2.21}$$

M_{mix} is the molecular mass of the mixture and has the following expression

$$M_{mix} = \sum_{i=1}^{N} M_i X_i. \tag{2.22}$$

Other terms in the above expression that have not been introduced are X_i, which is the species mole fraction and $\mathcal{D}_{ij}$ which is the binary diffusion coefficient $[A]$, $[B]$ and $[D]$ are $(N-1) \times (N-1)$ sized matrices. $[D]$ is a matrix of generalized Fick's law diffusion coefficients.

The thermal diffusion coefficient expression comes from FLUENT[4] and is

$$D_{T,i} = -2.59 \times 10^{-7} T^{0.659} \left[\frac{M_i^{0.511} X_i}{\sum_{i=1}^{N} M_i^{0.511} X_i} - Y_i \right] \cdot \left[\frac{\sum_{i=1}^{N} M_i^{0.511} X_i}{\sum_{i=1}^{N} M_i^{0.489} X_i} \right]. \tag{2.23}$$

It is an empirically based formula that takes into account both the concentration of species as well as the temperature of the flow. It is a form of the Soret diffusion coefficient which acts to cause heavy molecules to diffuse less rapidly, and light molecules to diffuse more rapidly toward heated surfaces.[4]

This detailed diffusion calculation is generally only needed when the flow is laminar. Turbulent diffusion generally overwhelms laminar diffusion, thereby making detailed specification of laminar species diffusion properties in a turbulent flow inessential.[4] One investigation in this thesis is the calculation of a turbulent combustion simulation and consequently a turbulent diffusion coefficient is required. For turbulent flows the mass diffusion flux can be written as

$$\vec{J_i} = -\big(\rho D_{i,m} + \frac{\mu_t}{Sc_t}\big)\nabla Y_i. \tag{2.24}$$

Here, $D_{i,m}$ is the diffusion coefficient for species i in the mixture, μ_t is the turbulent viscosity and Sc_t is the turbulent Schmidt number.

3. Reaction Rate Expression

R_i from (2.15) is the net rate of production/destruction of species i by all chemical reactions being modeled. Many different models exist to compute the reaction rate, R_i. The situation is similar in a sense to turbulence models. Many different models, of varying complexity, have been created and different things work in different situations. There are different layers of complexity of models depending on what level of accuracy is needed in the simulation of a combustive flow. One such model is the laminar finite-rate model[4] found within FLUENT.

A one-step chemical reaction of arbitrary complexity can be represented by the following stoichiometric equation:

$$\sum_{i=1}^{N_s} \nu_i^{'} S_i \rightarrow \sum_{i=1}^{N_s} \nu_i^{''} S_i. \tag{2.25}$$

S is an arbitrary specification of the chemical species, $\nu_i^{'}$ and $\nu_i^{''}$ are the stoichiometric coefficients for the reactants and products, respectively, and N_s is the total number of chemical species in the one-step reaction. An example which shows this notation is written as:

$$CH_4 + 1.5O_2 \rightarrow CO + 2H_2O, \tag{2.26}$$

where

$$\begin{array}{llll} S_1 = CH_4, & S_2 = O_2, & S_3 = CO, & S_4 = H_2O, \\ \nu_1^{'} = 1, & \nu_2^{'} = 1.5, & \nu_3^{'} = 0, & \nu_4^{'}=0, \\ \nu_1^{''} = 0, & \nu_2^{''} = 0, & \nu_3^{''} = 1, & \nu_4^{''}=2. \end{array}$$

A common notation in literature is to define the generalized stoichiometric coefficient as the difference between the stoichiometric coefficient of the product and the

reactant, or

$$\nu_{i,r} = \nu_{i,r}^{''} - \nu_{i,r}^{'}. \tag{2.27}$$

A generalized stoichiometric coefficient is defined for each species, i, in each reaction r.

From (2.15) the net mass rate of production of species i by all chemical reactions in the simulation is written as R_i. Its expression is written as the molecular mass of a certain species i, multiplied by the sum of the Arrhenius molar reaction rate of production/destruction of species i over all the reactions of which it is present, or in mathematical terms is

$$R_i = M_i \sum_{r=1}^{N_R} \hat{R}_{i,r}. \tag{2.28}$$

M_i is the molecular mass of the species i and N_R is the number of reactions that species i is present in. $\hat{R}_{i,r}$ is the Arrhenius molar rate of creation or destruction of species i in reaction r. It is important to re-emphasize that the subscript i denotes which species is being affected, and the subscript r describes in which reaction that species is being created or destroyed.

The molar rate of creation/destruction of species i in reaction r is given by [4,6]

$$\hat{R}_{i,r} = \left(\nu_{i,r}^{''} - \nu_{i,r}^{'}\right)\left(k_{f,r} \prod_{j=1}^{N_r} [C_{j,r}]^{\eta_{j,r}^{'}}\right). \tag{2.29}$$

This expression introduces many new terms. They are defined as follows:

N_r - number of chemical species in reaction r

$C_{j,r}$ - molar concentration of species j in reaction r.
Typical units are $\left[\frac{kmol}{m^3}\right]$

$\eta'_{j,r}$ - forward rate exponent of species j in reaction r

$k_{f,r}$ - forward rate constant for reaction r

$\nu'_{i,r}$ - stoichiometric coefficient for reactant i in reaction r

$\nu''_{i,r}$ - stoichiometric coefficient for product i in reaction r

It is important to note that this representation of $\hat{R}_{i,r}$ does not include the net effect of third bodies on the reaction rate; but, they can be added when third body reactions must be modeled. Also, the expression shown does not include backward reactions. This is because all simulations performed in this research modeled the backward reaction as another forward reaction. This was done so that the backward reaction could be assigned its own empirical reaction model within FLUENT.

The forward rate constant for reaction r, $k_{f,r}$, is computed using the Arrhenius expression[6,4]

$$k_{f,r} = A_r T^{\beta_r} e^{-E_r/RT} \tag{2.30}$$

where

A_r - pre-exponential factor

β_r - temperature exponent (lies between 0 and 1)

E_r - reaction activation energy

R - universal gas constant

This concludes a short description of the governing equations necessary to perform the simulations in this research. In addition to the governing equations many

expressions were introduced for important terms found within the governing equations. The next chapter will focus on the numerical techniques used to solve the governing equations with FLUENT.

CHAPTER III

NUMERICAL MODEL

In the previous chapter the equations that are necessary to capture the physics of the problem were given. In this chapter, additional information will be given regarding the numerical methods used to solve the governing equations. Also, an introduction will be given as to how the moving and deforming grid is implemented. For all flow simulations the computer software program FLUENT was used.

A. Description of Solution Method

One method to solve the fundamental fluid dynamic equations given in the previous chapter is a segregated or pressure-based technique. A segregated technique does not solve all of the governing equations at once, instead it solves them in a series of steps. Each step is outlined below:

1. An initial solution is given, or the most recently calculated flow properties are stored in the cells.

2. The three momentum equations are solved. Each momentum equation is solved individually based on the most current values for the remainder of the flow properties.

3. A "Poisson-type" of equation is solved to find a pressure distribution that automatically satisfies the continuity equation. This is necessary because the velocity distribution calculated in the second step does not automatically satisfy the conservation of mass. Usually a few sub-iterations are performed on the "Poisson-type" equation. Additional details will be given when the PISO pressure-velocity correction scheme is discussed.

4. Transport equations for scalar quantities such as turbulence kinetic energy, turbulent dissipation rate, energy, and species mass fractions are solved in turn using the previously updated values of the other variables.

5. A check for convergence is performed.

Convergence criteria is defined by the user. The governing equations are fully-coupled equations. In order to solve for all of the flow properties several iterations of the process outlined above must be performed. How many depends on set of convergence criteria which have been imposed by the user.

1. Linearization

For all of the work presented herein, the nonlinear governing equations are linearized with respect to the dependent variable of interest. This results in an algebraic equation for each transport equation for every cell in the domain. The unknowns are the dependent variables of the transport equations that must be solved. For example, if the transport equation is the species balance equation for carbon monoxide then the species mass fraction is the dependent variable. The species balance equation is linearized for every cell within the domain to form a set of algebraic equations where the mass fraction of carbon monoxide is the only unknown.

Due to an implicit linearization scheme each equation has more than just the unknown from its cell. The equation also has unknowns from neighboring cells. This results in a system of equations which is solved simultaneously for all of the unknown quantities of a certain transport equation at once. A point implicit Gauss-Seidel linear equation solver is used in conjunction with an algebraic multi-grid method to solve the resultant system of equations for the dependent variable in each cell.[4]

FLUENT uses the Gauss-Seidel method because it is generally economical in

memory requirements. In addition, it is often faster in computing a solution when compared to a direct solution method because the coefficient matrix has many zeros.

In summary, the pressure based solution technique implicitly linearizes a governing equation to create a system of equations. It then solves for all unknowns at the same time and then moves to the next transport equation.

2. Discretization

FLUENT uses a control volume discretization method to express the governing equations at a given point, or discrete cell, within the domain. In order to apply this technique the first step is to discretize the entire domain into a collection of cells. This is done through grid generation.

Using the finite control volume approach the transport equations are written in integral form. The second step is to apply the integral form of the governing equations to each and every discrete cell or control volume within the domain. When the discretization is applied surface integrals are created to account for the fluxes entering and leaving through the surface boundary of the cells. Any surface integrals resulting in the integral form of the transport equation are approximated by the sum of the fluxes crossing the individual faces of the discrete cell. Examples of such terms include convective and diffusion flux terms. Once these two steps are complete, it is then time to perform the linearization to the discretized equation and solve the system of equations. The interested reader should see the FLUENT user's manual for an example of a scalar transport equation written in integral form and discretized using finite volumes.

FLUENT stores discrete values of the flow variable at the cell centers. However, face values are needed to obtain the expressions for the surface integral terms because they require the flux across all faces of a cell. In order to calculate the value of the

dependent variable at the face of a cell an upwinding spatial discretization scheme is used. Specifically, this research will use a second-order accurate upwinding scheme. The following section will discuss some the details concerning the second-order upwinding scheme used in FLUENT.

a. Second-Order Upwinding Scheme

The second-order upwinding scheme used within FLUENT calculates the face values by taking into account what is happening upstream of the discrete cell. For an arbitrary scalar quantity, ϕ, the value of ϕ at a face is calculated by

$$\phi_f = \phi + \nabla\phi \cdot \triangle\vec{s}. \tag{3.1}$$

$\triangle\vec{s}$ is a vector pointing from the upstream cell centroid to the centroid of the face. The gradient of ϕ is computed using the divergence theorem and is given by

$$\nabla\phi = \frac{1}{V}\sum_{f}^{N_{faces}} \tilde{\phi}_f \vec{A} \tag{3.2}$$

where $\tilde{\phi}_f$ is the average of ϕ from the two cells on either side of the face, f.

b. Time Discretization

Simulations with moving and deforming meshes require discretization of the temporal term in the governing equations. The temporal discretization used here is second order accurate. This means a time derivative of the unknown flow property is approximated with a finite difference approximation. If we again assign ϕ the value of an arbitrary dependent scalar quantity that is a function of time and space then

$$\frac{\partial\phi}{\partial t} \approx \frac{3\phi^{n+1} - 4\phi^{n} + \phi^{n-1}}{2\Delta t}. \tag{3.3}$$

For this research an implicit time integration scheme is used which means all of the dependent variables in a transport equation that have been spatially discretized are expressed at time $t+\Delta t$, or the future time level. Consider $F(\phi)$ the rest of the terms in the transport equation that have been spatially discretized. Expressing $F(\phi)$ at the future time level and solving for ϕ at the future time gives

$$\phi^{n+1} = \frac{4}{3}\phi^{n} - \frac{1}{3}\phi^{n-1} + \frac{2}{3}\Delta t F(\phi^{n+1}). \tag{3.4}$$

Many sub-iterations are performed before the solution is actually allowed to advance in time. This means the entire process of solving the transport equation is performed many times, and many intermediate values of ϕ are calculated before the simulation is allowed to advance in time. The current research found 20 sub-iterations to work quite well during unsteady simulations.

It was discovered that the simulations of combustion in this research are sometimes unstable. It is important to note that the instability is not caused by the time discretization method used in the simulation. The advantage of the implicit scheme is that it is unconditionally stable with respect to time step size.[4]

3. Pressure-Velocity Correction

When using the segregated technique, the velocities are first calculated by solving the momentum transport equations. However, it becomes necessary to use a "Poisson-type" of equation to resolve the pressure field within the domain and compute a velocity field that will satisfy the continuity equation. Many different pressure-velocity correction techniques are available, and the Pressure-Implicit with Splitting of Operators (PISO) approach is used here.[3]

a. PISO Pressure-Correction Scheme

When using an uncoupled procedure to solve the discretized unsteady Navier-Stokes equations, the PISO pressure-corrections scheme may be used. The PISO scheme decomposes the pressure-correction scheme into a predictor-corrector strategy.[10] The scheme may be applied to both compressible and incompressible forms of the Navier-Stokes equations.

The PISO scheme applied to an incompressible flow is outlined in the following steps:

1.) *Predictor step.* The first step is to calculate or predict the velocity at an intermediate future time level. Using an implicit unsteady form of the momentum equation it is discretized as shown:

$$\frac{\rho}{\Delta t}(u_i^* - u_i^n) = \frac{\partial P^n}{\partial x_i} + H(u_i^*) + S_i. \tag{3.5}$$

This discretization uses index notation where the superscript * represents an intermediate value of, in this case, velocity and the superscript ^{n} is the current value. Therefore, the intermediate value of velocity is written as a function of the current pressure distribution. $H(u_i^*)$ represents the spatial convective and diffusive fluxes of momentum calculated with the intermediate velocity. S is any source term in the momentum equation. This intermediate velocity does not necessarily satisfy the continuity equation. Therefore a corrector step is required.

2.) *Corrector step.* The first step in the corrector procedure is to calculate an intermediate pressure. From this intermediate pressure a new velocity is calculated which automatically satisfies the conservation of mass. Using an explicit and unsteady form, the momentum equation is written as

$$\frac{\rho}{\Delta t}(u_i^{**} - u_i^n) = -\frac{\partial P^*}{\partial x_i} + H(u_i^*) + \vec{S}_i. \tag{3.6}$$

The revised velocity for an incompressible flow must satisfy

$$\frac{\partial u_i^{**}}{\partial x_i} = 0. \tag{3.7}$$

in order to be in agreement with the physical equations.

Taking the divergence of (3.6) and substituting (3.7) gives the following form of the Poisson equation:

$$\frac{\partial^2 P^*}{\partial x_i^2} = \frac{\partial H(u_i^*)}{\partial x_i} + \frac{\partial S_i}{\partial x_i} + \frac{\rho}{\Delta t}\frac{\partial u_i^n}{\partial x_i}. \tag{3.8}$$

All terms on the right-hand side have already been determined. So the intermediate pressure is calculated and used in (3.6) to calculate u_i^{**} such that the conservation of mass is satisfied. The corrector step is then repeated, as Issa suggests that two correction steps are sufficient for most purposes.[10] This pressure-correction scheme is used because time-accurate solutions can be simulated without changing the physical time step used to advance the solution. Other pressure-corrections schemes sometimes require a smaller time step be taken during the pressure-correction portion of the numerical solution.

4. Moving Deforming Grid

Similar to turbulence modeling and chemical reaction modeling, there is more than one way to model a dynamic mesh. The dynamic mesh models used for this research are divided into two main sections. One model is called a spring-based smoothing method and the other is coined as a local remeshing method.

a. Spring-Based Smoothing Method

The spring-based smoothing method works by treating each of the line segments between two mesh nodes as a spring. All of these line segments then create a network of springs that are all connected together. The original mesh that is created has all springs in neither tension or compression. Therefore there is no force pulling one node away from another. Once a boundary of the domain begins to move the nodes on the boundary move with it and start to create forces on the nodes caused by the springs. If two nodes are too close then the spring force will act to repel those nodes away from each other, and if the nodes are too far away, the spring force pulls the nodes closer together. The placement of each node depends on all of the nodes surrounding it. Once the boundaries are moved, the neighboring nodes will move due to the spring forces until a new equilibrium position is found. Hooke's Law says the spring force is equal to the spring constant multiplied by the displacement of the spring. Each node has n_i number of nodes connected directly to it with springs. Therefore the total force on a node is

$$\vec{F}_i = \sum_{j}^{n_i} k_{ij}\big(\Delta\vec{x}_j - \Delta\vec{x}_i\big). \tag{3.9}$$

k_{ij} is the spring constant and is defined as

$$k_{ij} = \frac{1}{\sqrt{|\vec{x}_i - \vec{x}_j|}}. \tag{3.10}$$

$\Delta\vec{x}_i$ and $\Delta\vec{x}_j$ are the displacements of nodes i and j respectively.

For each node an equilibrium state must be found, meaning that the forces on a node must sum to zero. First the boundaries are displaced and then an iterative equation is used to find the displacement of all of the interior nodes. The iterative equation is expressed by

$$\Delta\vec{x}_i^{m+1} = \frac{\Sigma_j^{n_i} k_{ij} \Delta\vec{x}_j^m}{\Sigma_j^{n_i} k_{ij}}. \tag{3.11}$$

This equation must be iterated over all the cells in the interior of the domain. The sweep through the cells acts as a smoother inasmuch as it finds the location of one node based on all of the other nodes around it by averaging, and then performs this for each node in the interior of the domain. The new node locations at the next time step are

$$\vec{x}_i^{n+1} = \vec{x}_i^n + \Delta\vec{x}_i^{m+1,converged} \tag{3.12}$$

where $\Delta\vec{x}_i^{m+1,converged}$, is the value of $\Delta\vec{x}_i^{m+1}$ once the movement is less than a specified amount set as the convergence criteria. In order to update to the new node locations, all of the nodes within the domain must move a specified amount that is under the convergence criteria that has been set.

One of the main advantages of the spring-based smoothing remeshing technique is that the number and ordering of nodes and lines does not change with deformation of the grid. However, this only applies when the displacements are small relative to the size of the local cells. If displacements become too large then cell skewness can be affected, creating inadequate cells. Thus spring-based smoothing is only sufficient at some instances. Another technique is needed for large deflections.

b. Local Remeshing Method

In terms of computational expense, it is generally desirable to keep the same amount of nodes and cells in any grid. This is because information about the grid does not need to be updated at every time step if the same node numbering is preserved.

However, there are instances in dynamic mesh modeling where cells move and become highly skewed or even inverted. In these instances, it is necessary to remesh a certain region of cells.

The basic idea behind the local remeshing method is to evaluate the new cells after spring-based deformation. Certain cells are marked for remeshing if they are smaller than a specified minimum size, larger than a specified maximum size or if the cell skewness is greater than a specified maximum cell skewness.[4] If cells are found which do not meet these criteria then these cells are remeshed. This technique is currently only valid for triangular cells in two dimensions in FLUENT. The cell height is the parameter which is responsible for controlling remeshing. If the cell is expanding it is allowed to expand until

$$\delta > (1 + \alpha_\delta)\delta_{ideal}. \tag{3.13}$$

Here α_δ is a height factor set by the researcher depending on the problem being simulated. The ideal height is the height of the cell when it is originally created. On the other hand, if a cell is shrinking it may shrink until

$$\delta < \alpha_\delta \delta_{ideal}. \tag{3.14}$$

Typically, bad cells appear in conglomerate regions such that remeshing is easier. A hole in the mesh is created by removing cells which were unfit to reuse. Then this hole is remeshed with new cells which typically are similar in edge length to surrounding cells by using a grid generator algorithm. The new cells are checked to ensure cell quality of the local remeshed region. If needed, it is possible to override minimum cell sizes if this is found to be the only way cells can be remeshed with satisfactory skewness levels. New cells are assigned new variable values based on old cell variables

and neighboring cell values.

CHAPTER IV

RESULTS FOR MOVING DEFORMING MESH

FLUENT proclaims that an object can be moved around while the flow solution is computed at each time step. This capability is an excellent start to studying fluid-structure interaction. However, this capability needs to be verified and explored. Documentation does not fully describe how these problems should be solved nor does it provide examples of how to solve such problems. Instead the nuts and bolts are described and it is left up to the user as to how they are implemented.

This chapter will begin by discussing how the moving and deforming mesh operates within FLUENT. First a discussion will be devoted to the grid generation which allows for a moving body. A user defined function (UDF) must be defined to activate movement of a body within a mesh. An example of a user defined function necessary to do this is given. A simple vertical sinusoidal movement of a circular cylinder is shown. Next an unsteady flow solution is performed to test how well FLUENT can capture the shedding frequency of a circular cylinder. Finally, a circular arc is assigned the motion of a hornet insect wing, and a combination of moving and deforming a rigid body while simultaneously solving for the flow around the body is performed.

A. Grid Generation for Moving Deforming Mesh

When enabling the moving deforming mesh, the grid must be built in a certain manner. Grid creation was performed with the help of the software grid generator GAMBIT. This software is sold in the same package as FLUENT and works well for the creation of moving deforming meshes. Even though GAMBIT is set to operate with FLUENT specifically, third party grid generation software such as GRIDGEN works

just as well.

When creating a grid for moving and deforming usage, some additional thought should be placed into how it is created and how it should be created depends on why and how the object will be moved. As mentioned in chapter III, FLUENT offers a few different options depending on the magnitude of the displacement of the body. The two options which were used are the spring model and the remeshing model. A brief recap of each model is given below:

- Smoothing: Interior nodes behave as if they have a series of springs attached to them. This enables the nodes which define the cells to be squished or pulled but the same number of nodes and cells remain. Thus connectivity remains the same. This method works well, but only when the displacement of the boundaries is relatively small compared to the distance between the nodes on the same boundary.

- Remeshing: Creates new cells when the skewness of old cells becomes too large. Remeshing is better than smoothing for objects that move large distances in any direction but it is computationally more expensive than smoothing because new connectivity is needed after every remeshing. This technique is so far limited only to two-dimensional (2D) triangular cells.

An important note is to say that a combination of these grid-deforming tools was used for all calculations performed.

If all that is needed is to move the boundary of a solid body, such as a cylinder, then little planning is needed in creation of the grid. If instead, a boundary layer grid created around the circular cylinder, in an effort to resolve viscous behavior, is required to move with the cylinder, then more care must be taken. The best way to ensure that the boundary layer mesh will move with the cylinder is to create the

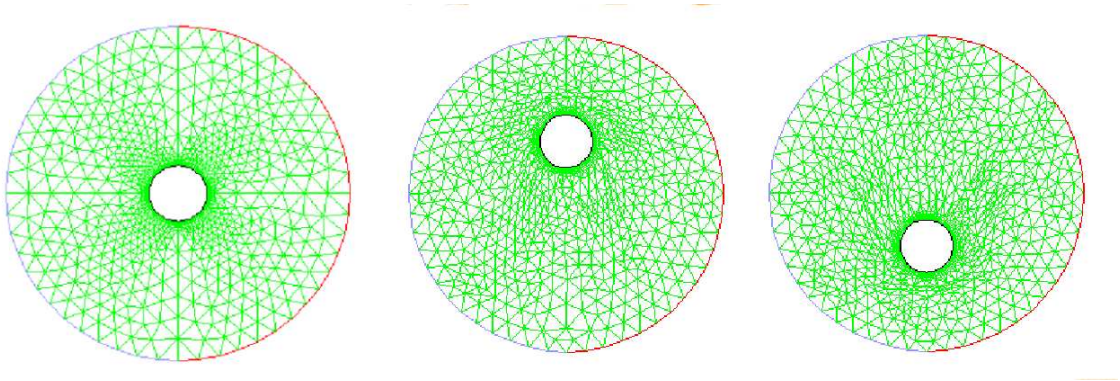


Fig. 1. Sample of moving deforming mesh. Picture at far left is the initial grid, middle picture shows the cylinder when it has reached its peak displacement upward and the far right picture shows the cylinder at the bottom of its translation.

boundary layer grid as its own face in GAMBIT. GAMBIT will then allow the users to define it as a specific zone when it is exported to FLUENT. Inside FLUENT, this zone, and only this zone will be picked to move in the manner in which the user prescribes. There are some ways already built into FLUENT in which a user can move a wall or give a body a fixed velocity but for more complicated movements a user defined function (UDF) is needed. Such a UDF was written to define the motion of a cylinder to be a sine function oscillating in the vertical direction with amplitude equal to the diameter of the the cylinder. Figure 1 shows a few snapshots of the grid as the cylinder is moved.

B. User Defined Functions

Thus far the discussion of moving deforming meshes has been only with regard to the creation of the grid, and the methods used to deform the grid. This section presents the manner in which the movement is prescribed.

A UDF allows many different avenues for a user of FLUENT to change or add capabilities to their simulation. Whether it be adding a transport equation, changing a transport equation, redefining boundaries, or changing material properties the user

is given some authority to change or define quantities by writing their own UDF. The UDF used here allows the user to define the motion of a body within a mesh. This is done by programming a UDF (basically a subroutine) in the C programming language. Initial programming of these functions can be difficult because programmers new to the C language may have a hard time distinguishing C commands from pre-packaged functions in FLUENT called macros. Unfortunately, the documentation of many of the pre-packaged functions (or macros, as FLUENT calls them) is not complete. If the moving of a body can be modeled as a rigid body motion, then the macro DEFINE_CG_MOTION should be used. The author has used this particular define macro and found it to work quite well when prescribing the motion of a rigid body. Figure 1 shows an example of a cylinder moving in a sinusoidal vertical motion which uses the DEFINE_CG_MOTION macro.

The macro that allows for the movement of node positions individually is called DEFINE_GRID_MOTION. If fluid-structure interaction were to be done around a deformable body this would be the macro that would allow the user to update the new nodal coordinates based on the deformation that had been determined. Care must be taken to prevent cells from overlapping in one time step. If a boundary nodes movement is greater than the distance between nodes at the boundary, FLUENT will simply crash because the cells have inverted and become invalid. This predicament becomes especially important when a fine boundary layer mesh is constructed. In order to solve a moving boundary simulation with appropriate boundary layer clustering, the user must use extremely small time steps so as to not invert a boundary layer cell. This causes an increase in time necessary to solve the problem. An option that allows for a larger time step is to create a course mesh with no boundary layer clustering, but this option comes with the trade off of resolution of the boundary layer effects.

To use the UDF in FLUENT the user saves their UDF with a ".c" extension. It is recommended to write the UDF with an application such as notepad and then save the file as something similar to "filename.c." Then, while FLUENT is open but idle, the user can choose to either interpret or compile the code. It is recommended that all UDFs be compiled. After selecting the compiling option the user can add a source file. If the source file (filename.c) is in the directory FLUENT was launched then the user can select the source file and then build it. Upon using the build command a list of makefile commands, warnings, and possibly errors will display on the FLUENT screen. Careful consideration should be taken when warnings and errors occur. Then the user selects to link the source file to FLUENT. If this command is successful then the source code has passed the compilation stage and is ready to be used in FLUENT. Unfortunately passing compilation does not ensure a perfect working UDF.

As mentioned previously, UDFs are not only applicable to moving and deforming meshes but can also be applied to specify boundary conditions and even solve a different version of the energy equation. An example of another type of UDF is discussed in chapter VI. To link a deforming mesh UDF one uses the options Define $\Rightarrow$ Dynamic Mesh $\Rightarrow$ Zones. Then the user must select the zone and assign it the UDF that is listed. If only one UDF was compiled then only one option should appear under the Motion UDF/Profile category. For more information about troubleshooting and setting up UDF's the reader should refer to the FLUENT UDF manual.[11]

An example of a UDF is shown in figure 2 which was written to define the motion of a cylinder to be a sine function in the vertical direction. This movement is similar to the motion shown in figure 1 except with only a half diameter amplitude rather than a full diameter. The DEFINE_CG_MOTION macro was used. The input and output parameters of this UDF are the following:

up-down:	Name that appears in FLUENT to help user select UDF
dt:	"Dynamic Thread", where a thread is defined as a sub-region, or a smaller collection of cells from the entire domain. For example, a thread could be the collection of cells that make up a boundary layer mesh around a body. That way FLUENT knows which particular cells to assign the velocities to.
cg_vel:	The output of the translational velocity from the UDF
cg_omega:	The output of the angular velocity
time:	Time, in seconds, of the flow solution
dtime:	Time step the user specifies

C. Low Speed Unsteady FLUENT Solution Investigation

In order to verify the solution of an unsteady flow, FLUENT was used to solve for the flow around a circular cylinder. A circular cylinder was used because it is well known how a laminar flow around a circular cylinder behaves. The flow behind the cylinder becomes unstable; the vorticies are alternately shed from the body in a regular fashion and flow downstream.[12] This circular cylinder problem is not too far removed, in an aerodynamic sense, from the the problem of a flapping airfoil. If FLUENT is able to correctly capture the unsteady effects of a circular cylinder, then it shows promise for its ability to capture the aerodynamic phenomena that would result from a flapping motion, provided that the time step used to advance the numerical solution is small enough to capture all of the unsteadiness.

The left side of figure 3 shows the whole domain that consists of 49,072 cells. The little spot in the middle is the actual circular cylinder. One conclusion reached through numerical experiments was that FLUENT does not have sufficiently adequate

```
/*UDF which defines a simple sine function to a rigid body in the y-direction */

/*   AUTHOR: Steven Chambers  */
/*   Date  :  July 15, 2003   */

#include <stdio.h>            /*Input-output header file */
#include "udf.h"             /*Needed in every UDF      */
#include "dynamesh_tools.h"   /*Used to ensure mesh motion */
#include <math.h>             /*Included to ensure sin and cos functions operate
 */

static real loc, vel;          /* Declaration of variables */
/* static is used to ensure that other functions outside source cannot use loc a
nd vel.*/
DEFINE_CG_MOTION(up_down, dt, cg_vel, cg_omega, time, dtime) /*UDF macro declara
tion */
{
#if 1
  FILE *fp;   /*Assigns a pointer from file to fp */
#endif

/* reset velocities */
  NV_S (cg_vel, =, 0.0);
  NV_S (cg_omega, =, 0.0);

  if (!Data_Valid_P ())
    return;               /*Exits UDF is memory is not allocated */

  loc = 0.5*sin(time*M_PI/5.0); /*Defines the location of the CG wrt initial pos
ition*/
  vel = M_PI*0.1*cos(time*M_PI/5.0);   /*Defines velocity of CG*/

#if 1
  if( (fp=fopen("out.dat","a")) !=NULL )/* Writes out data to an output file out.d
at*/
    {
      fprintf(fp,"%f  %f  %f\n",
              time,vel,loc);
      fclose(fp);
    }
#endif

/* set cylinder velocity */
  cg_vel[1] = vel;         /*Sets y-velocity equal to velocity defined above*/
}
```

Fig. 2. Sample of a user defined function (UDF) that defines a vertical sinusoidal movement to a cylinder.

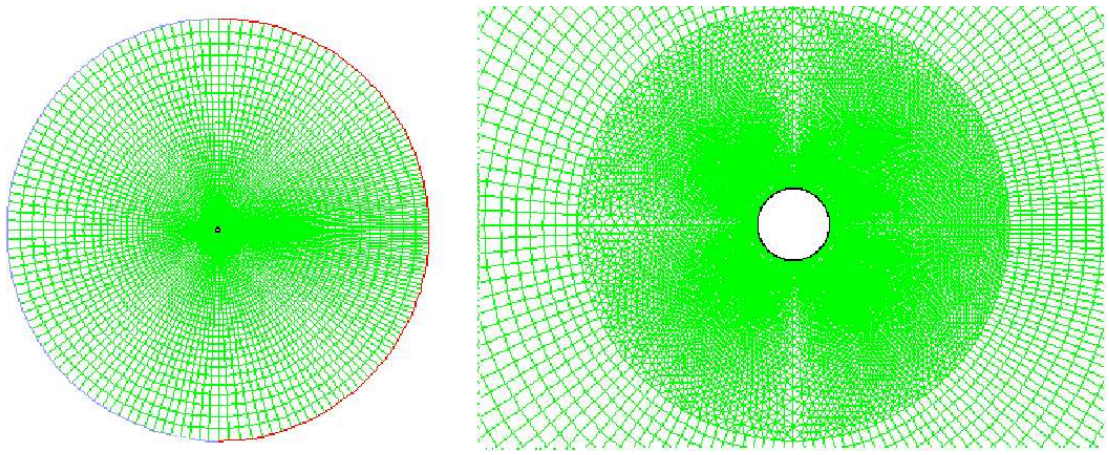


Fig. 3. Outer domain of circular cylinder mesh.

non-reflective far-field boundary conditions. As a result, this grid has boundaries placed at 50 diameter lengths away. It was hoped that with the boundaries so far away, the solution next to the cylinder would be minimally affected by the boundaries.

Also, to take advantage of the moving deforming mesh capabilities found in FLUENT, it must be possible to invoke the remeshing capabilities. Because remeshing only works with triangular cells in two dimensions, and to limit the total number of cells in the domain, the discretized space was broken into three sections. The outer most section is a structured grid, and then there is a middle portion, which is shown at the right of figure 3, that is comprised of triangular cells. Figure 4, shows the inner most grid, which is structured. This structured grid next to the cylinder provides good control over the cell size in order to properly capture the flow variation in the boundary layer.

The thickness of the boundary layer structured grid was estimated by using Thwaites method. The equations used are shown below.

$$\overline{U_e} = \frac{U_e}{U_\infty} \tag{4.1}$$

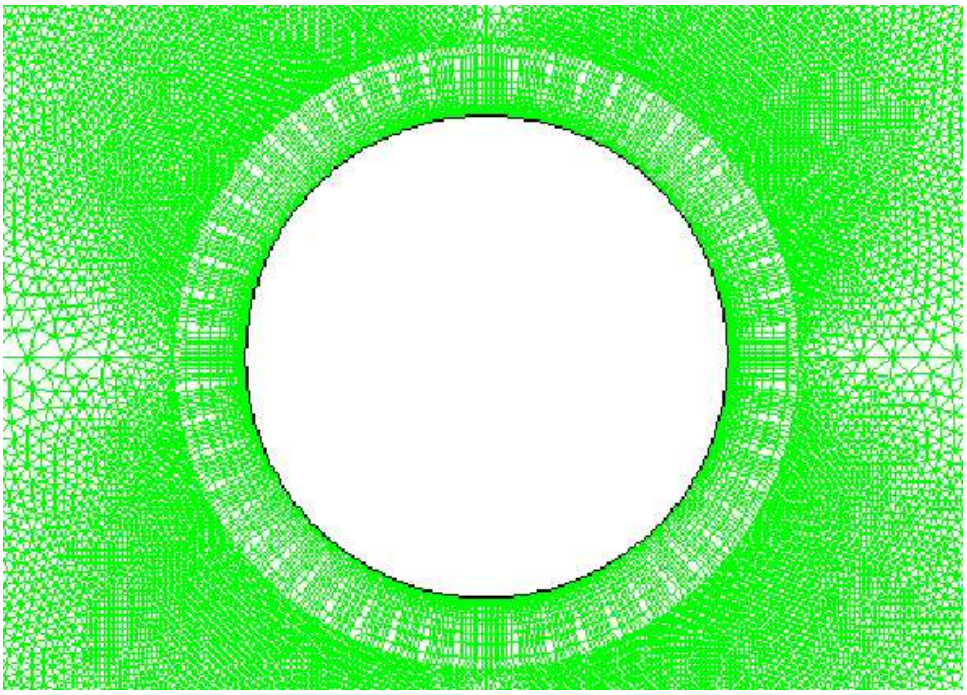


Fig. 4. Boundary layer mesh surrounded by unstructured grid.

$$\overline{U_e} = 2\sin(x^*), \quad \text{where} \quad x^* = \frac{x}{r} \tag{4.2}$$

$$\Lambda = \overline{\delta}\frac{d\overline{U_e}}{dx^*}, \quad \text{where} \quad \overline{\delta} = \frac{\delta}{2r}\sqrt{Re_D} \tag{4.3}$$

In the equations above U_e is the velocity at the edge of the boundary layer, x is the circumferential length from the leading edge stagnation point, r is the radius of the cylinder, and δ is the dimensional boundary layer thickness. The assumptions made are that the edge velocity is taken from potential flow, the boundary layer is thickest where it separates and that the separation point on a circular cylinder, when using the potential flow solution as the edge velocity with Thwaites method, is found to be 104.5° from the leading edge stagnation point. From this the derivative of the edge velocity at separation can be approximated. Also knowing Λ is approximately -8 to -12

when separation occurs allows for the computation of the non-dimensional boundary layer parameter. Lastly, knowing the Reynolds number of the flow, a dimensional boundary layer distance at separation can be approximated. This method was used to control the number of points that exist inside the boundary layer mesh. The boundary layer thickness found was then compared to Thwaites approximation for a flat plate to ensure reasonable results. The boundary layer for a Reynolds number of 500 was estimated to be 0.123 m. This was used as the height of the boundary layer zone with approximately 18 points inside with a geometric growth factor of 1.2.

Ultimately it was hoped to model this circular cylinder as an elastically mounted cylinder. In fact, that was the driving reason of generating the grid in the manner in which it was created. The physics of an elastically mounted cylinder have been explored both numerically and experimentally and the flow conditions used for all calculations shown were obtained by matching Reynolds number for these cases. [13,14] The references solved this problem non-dimensionally, however, FLUENT solves the governing equations dimensionally. Thus the parameters shown in table I were used to match the Reynolds number used in the references. [13,14]

The grid shown was then used in combination with the parameters from table I to provide flow boundary conditions. The cylinder was modeled as a no-slip wall boundary condition. The inlet was modeled as a velocity inlet boundary condition where the inlet velocities and components are provided. The inlet velocity was 7.3e-03 m/s in the x-direction only. There is no y-component of velocity assigned at the inlet. The velocity inlet boundary condition in FLUENT adjusts static pressure to accommodate prescribed velocity distribution.[4] Stagnation properties of flow can vary across the boundary, which can lead to non-physical results if velocity inlet boundary conditions are used for compressible flow.[4] Because this simulation is so far removed from being compressible, the velocity inlet boundary condition is valid.

Table I. Flow parameters used in simulation

Parameter	Value	Units
U_∞	7.3e-03	m/s
D	1.0	m
T_∞	288	K
ρ_∞	1.225	kg/m^3
P_∞	101327	N/m^2
μ_∞	1.789e-05	$Nsec/m^2$
Re_D	500	N.A.

Modeling the outlet of the domain was done by assigning the "outflow" boundary condition. Data at the exit plane are extrapolated from the interior and mass balance balance correction is applied at the boundary.[4] Flow exiting "outflow" boundaries exhibit zero normal diffusive flux for all flow variables.[4]

After initializing the flow-field to be the same as at the inlet, and implicit, segregated, second-order, unsteady, 2D, double precision, laminar solver was used to compute the time accurate solution. The discretization scheme was a second-order upwinding scheme for the momentum equations. Asymmetric vorticity shedding occurred after approximately 8 vortex shedding time periods (about 5000 *sec*). This was due to the fact that it took the solver some time to resolve the instabilities of the problem from the initial condition given. The solution was marched in time using a one second time step with 20 sub-iterations per time step. The majority of calculations were performed using 4 parallel processors. With the 4 processors each time step calculation was performed in approximately 20 seconds. By comparison, if the same job was given to a singe processor, one time step calculation took about 55 seconds.

This means the same calculation can be performed in almost a third of the time, by having 4 times the number of processors. The efficiency of the parallel computation was 69%. This scaling just represents one instance, and is not necessarily indicative of how all problems will scale.

Results for the rigidly mounted cylinder showed good agreement with published data.[13,14] The Strouhal number is a dimensionless parameter defined as,

$$St = \frac{fD}{U_\infty}, \tag{4.4}$$

where f is the frequency of vorticies shed in a vortex street, D is the length scale, and U_∞ is the speed of the fluid flow. Vorticies are shed when St is approximately 0.23 for flow at these conditions.[14] Using (4.4), at the current flow conditions, the corresponding time period is about 595 *sec*. Figure 5 shows that the time period is about 600 *sec*. Therefore, the frequency of vortex shedding is in good agreement with previous work.[14] An important note here is that this graph was created using a five-second time step with 20 sub-iterations per time step.

To further test FLUENT, the time step was increased to 10 seconds. Figure 6 shows the results of 4 additional cycles computed with this new time step. While the time period appears to remain unchanged the amplitude of the lift diminishes slightly with the increased size in time step.

Testing the number of sub-iterations that were necessary to capture the shedding vortex phenomena required further testing of FLUENT. At first, the step size was returned to 5 *sec* per time step. Then the number of sub-iterations was set to 15. With this setup the results seemed unchanged. Further decreasing the number of sub-iterations to 10 resulted in a sharp decrease in the quality of the results. Figure 7 shows the non-dimensional lift vs. time, where the first two cycles where computed using 5 second time steps with 20 sub-iterations. The next four cycles were computed

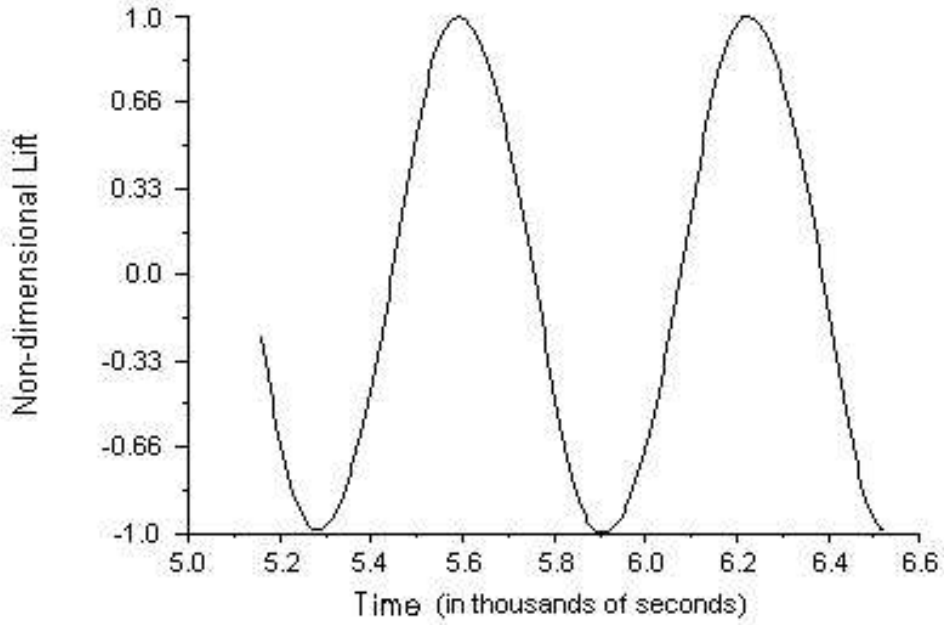


Fig. 5. Non-dimensional lift versus time for 5 *sec* time steps.

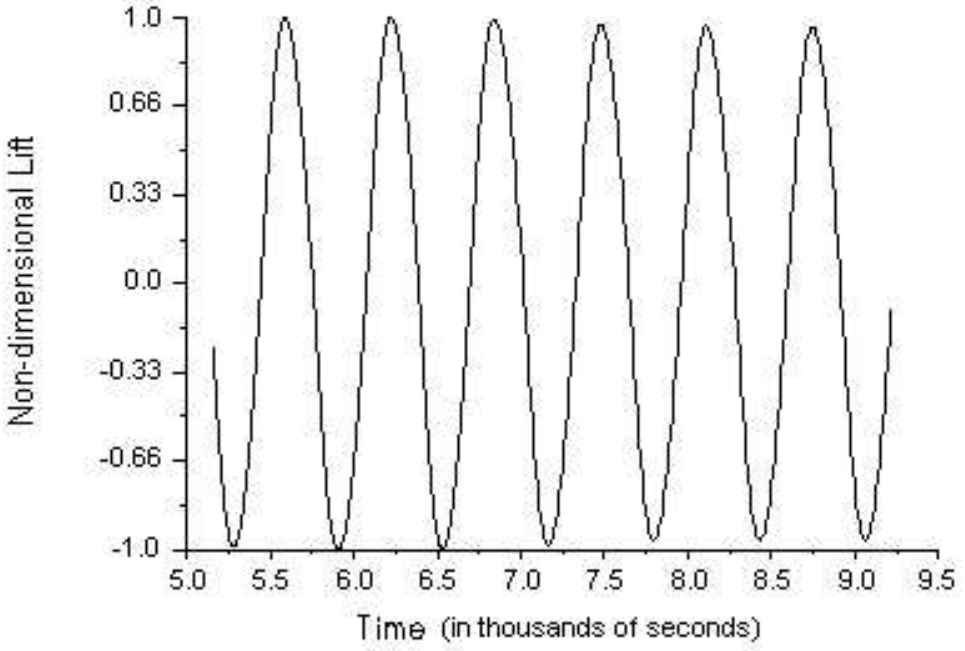


Fig. 6. Non-dimensional lift versus time for 10 *sec* time steps.

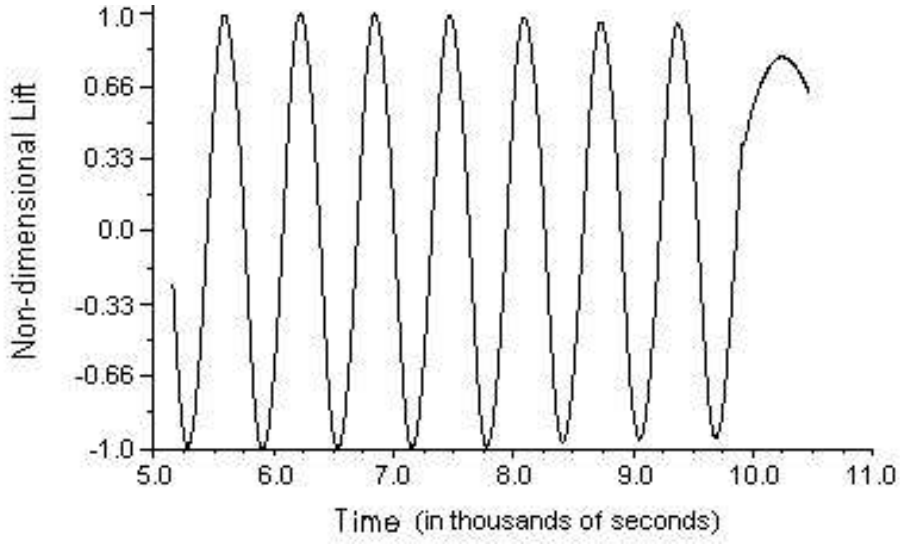


Fig. 7. Non-dimensional lift versus time for 10 *sec* time steps with 10 sub-iterations.

using 5 second time steps with 15 sub-iterations. The last cycle shows what happens when the number of sub-iterations was set to 10, while continuing to use a 5 second time step.

Figure 8 shows the effects of the domain resolution on the solution that is obtained. The left portion of figure 8 shows great detail in a snapshot of vorticity magnitude. The associated grid that is used to obtain the snapshot is shown immediately below. The contour at the right of figure 8 shows the same snapshot, at the same instant, zoomed out. Its corresponding grid is shown below as well. As the number of cells lessens or as the grid loses its refinement the results seem to lose their clarity. The poor resolution of vorticity further down the wake is most likely due to a course grid along the cylinder wake. It can be seen that the cell sizes in the wake are relatively large where shed wake vorticies seem to smear and become less pronounced. If vortex shedding downstream is what is of interest then a different grid should be made that will better capture the asymmetric pattern for a longer distance down the wake.

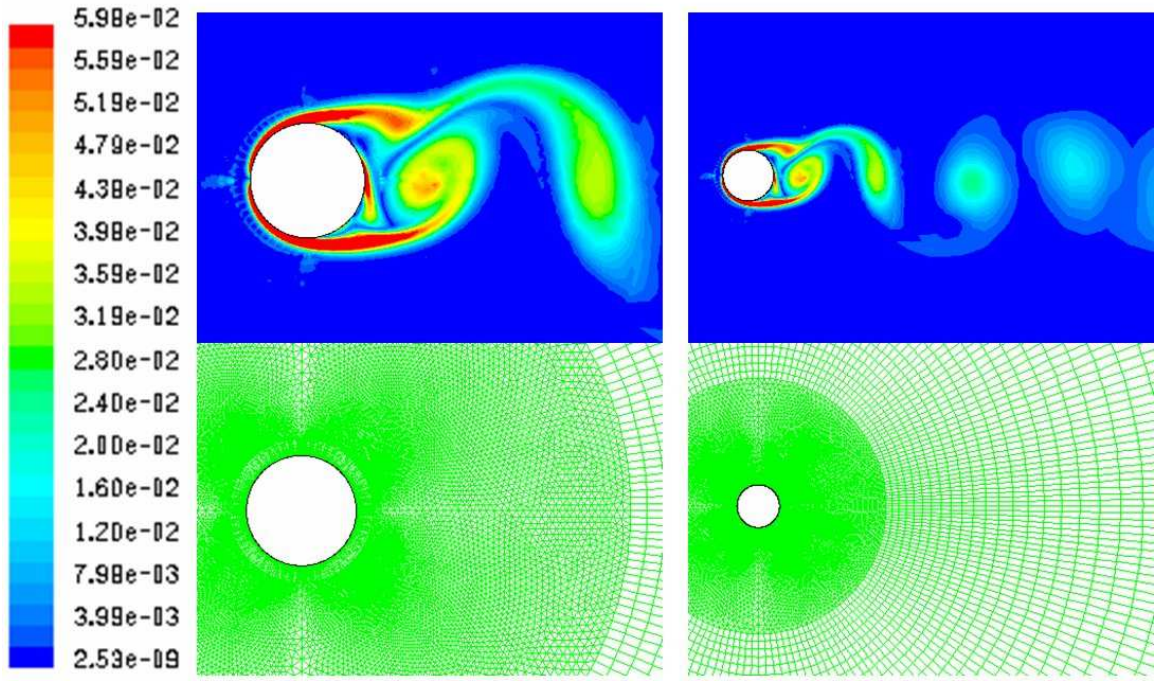


Fig. 8. Vorticity magnitude and grid resolution.

The unstructured region of this grid was originally given a 3.5 diameter radius for use in the elastically mounted cylinder problem. An elastically mounted cylinder is a dynamic problem where a circular cylinder has a network of springs and dampers attached to it and as the vorticies are alternately shed the circular cylinder is allowed to move due to the asymmetric pressure distribution caused by the unsteadiness of the flow. The cylinder displaces 1.5 diameters in the positive and negative y-direction (vertical direction) and about 1 diameter in the positive x-direction.[13] These displacements depend on spring and damper constants. It is hypothesized that this mesh can still be used in the initial calculation of the elastically mounted cylinder, but a mesh that has more resolution in the wake region would better compare with previous work.

From the results shown, it has been determined that FLUENT has the capability to capture unsteady aerodynamic phenomena, depending on the level of grid

resolution and the time step that is used.

D. Application of Moving Deforming Mesh to Flapping Flight

Another stage in verifying FLUENT's applicability to fluid-structure interaction problems is to test its moving deforming mesh capabilities. The circular cylinder displacement showed that a rigid body can be moved, but it did not show whether inadequate grid skewness and inversion might occur as the simulation is performed. For this problem, a small circular arc was chosen to model and insect wing. This arc would then be assigned a flapping motion of an insect such that the deforming mesh capability within FLUENT could be tested in an interesting problem.

The arc is 2D, has a chord length of approximately 6 cm, and a thickness of 1 mm. The main emphasis placed while creating this grid was not so much on grid resolution of the shed vorticies as it was on having a grid size of a manageable magnitude to test out the different remeshing capabilities and get a rough idea of some of the aerodynamics which result. The boundary of the the entire domain is a rectangle which has its edges about 7 chord lengths away in the upstream and downstream directions and about 5 chord lengths away on the top and bottom. This grid was broken into two zones. The first is shown in figure 9.

Figure 9 shows the inner zone. In an effort to resolve the boundary layer as the wing flaps, a structured grid was placed around the arc and then moved in the same manner as a hornets wing flaps. The thickness of this boundary layer grid was obtained by using Thwaites method for boundary layer over a flat plate. The main reason this inner zone was created was to allow the boundary layer to move in the same prescribed flapping motion as the arc itself. Before a zone can be moved it must be declared as a movable zone in FLUENT. The remaining part of the face was filled

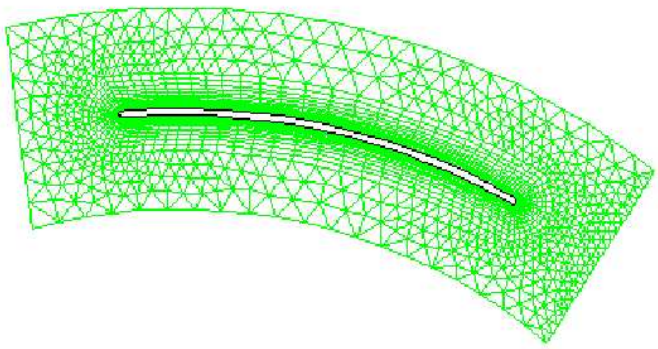


Fig. 9. Moving portion of arc grid.

in with triangular elements to control the number of points in and around the airfoil. It is this entire inner zone (figure 9) that moves in a prescribed manner throughout the larger domain.

The flapping motion of the arc was taken from that of a forward flying hornet.[15] This reference gives information about the angular and translational velocities of the hornet wing over time. These velocities were applied to the arc grid described above through the use of a UDF. More specifically, the velocity of the inner zone was prescribed using the FLUENT macro DEFINE_CF_MOTION. Figure 10 shows a reproduction of the angular and translational velocities of a forward flying hornet where the time and amplitude are non-dimensional.[15]

Figure 11 shows the moving deforming mesh for the prescribed motion. The top right-hand portion of figure 11 shows the initial grid. The top left shows the arc at the bottom of the stroke. The lower right figure shows the arc during its upstroke. The lower right portion of figure 11 shows that as the arc travels upward, a trail of points is left at the back of the arc. The bottom left figure shows the arc returned to its initial position. Here the full trail of fine grid points is seen. The trail of grid points is a result of the rotation of the arc as it moves vertically. If more care had

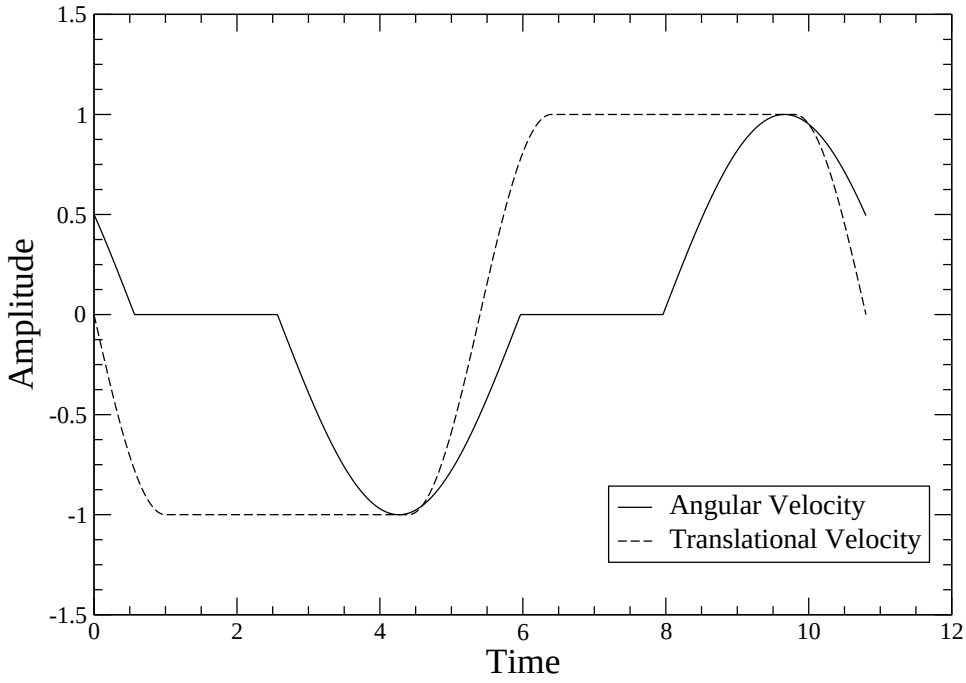


Fig. 10. Translational and angular velocity of forward flying hornet.

been taken in how the grid was created, then the trail of fine grid points could have been eliminated. Also, it should be noted that this series of slides makes no attempt at using grid adaptation. This particular research instead decided to focus on the ability for a FLUENT user to move the mesh while at the same time preserving some of the mesh refinement desired when the initial grid was created. Flow conditions were not taken from a pre-existing problem known to the author. It was known that the customer was interested in running the problem at a Reynolds number of 5000. Based on this Reynolds number, table II shows what flow conditions used in the FLUENT simulation.

The arc was modeled as a no-slip wall boundary condition. The remaining boundary conditions were defined along the outer edge of the rectangular domain. The inlet was modeled as a velocity inlet boundary condition where the inlet velocity magnitude and direction are provided. The inlet velocity was 1.24 m/s in the x-direction

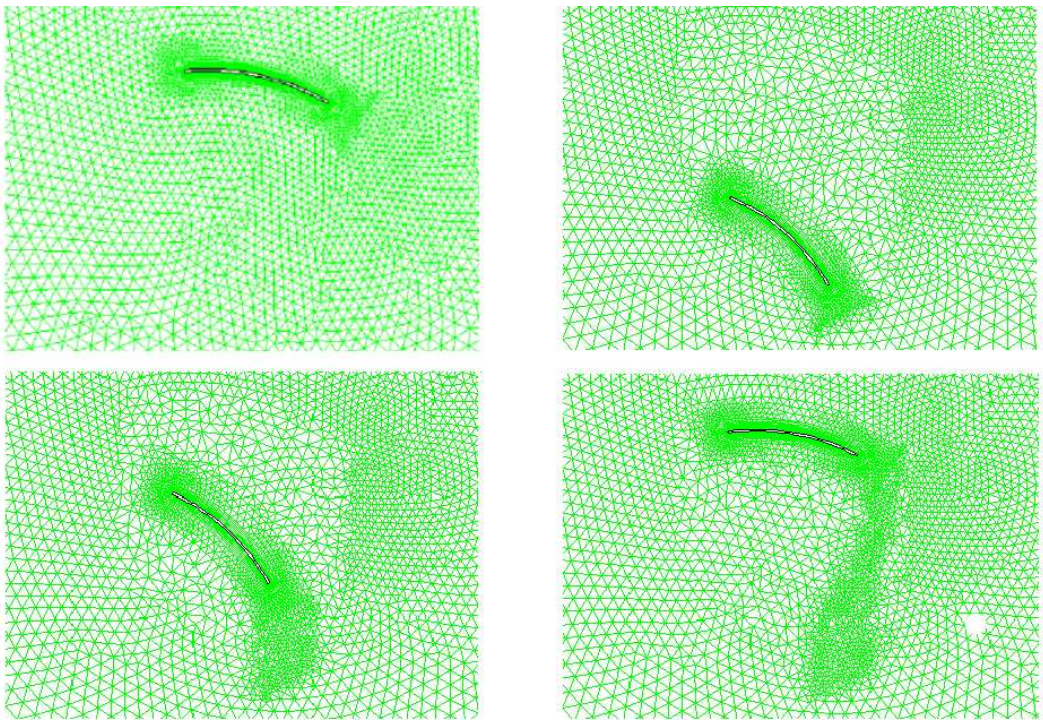


Fig. 11. Mesh plots showing grid resolution during flapping motion.

only. Similar to the cylinder problem, the flow conditions are incompressible and the velocity inlet boundary condition is appropriate.

Modeling the outlet of the domain was done by assigning what FLUENT calls an "outflow" boundary condition. This boundary condition requires no pressure or velocity information, instead data at the exit plane are extrapolated from the interior. This boundary condition was the same boundary condition as was specified in the cylinder simulation. The top and bottom of the domain were modeled as "outflow" boundaries as extrapolated values from the interior were believed to have a small amount of influence on the solution and therefore mimic far-field boundary conditions adequately.

After initializing the flow-field to be the same as at the inlet, an implicit, segregated, second-order, unsteady, 2D, double precision, laminar solver was used to compute the time accurate solution. Whenever mesh motion is required, the problem

Table II. Flow parameters used for flapping arc simulation

Parameter	Value	Units
U_∞	1.24	m/s
c(chord)	0.0587	m
T_∞	288	K
ρ_∞	1.225	kg/m^3
P_∞	101327	N/m^2
μ_∞	1.789e-05	$Nsec/m^2$
Re_D	5000	N.A.

must be solved using the unsteady solver. This is an obvious requirement as if the geometry is changing with time then the solution will be unsteady. The discretization scheme used was a second-order upwinding scheme for the momentum equations with a PISO scheme for pressure-velocity coupling. The specifics of these methods are discussed in chapter 2. All calculations for this simulation were performed on a single processor because many times a UDF must be parallelized before it can be used on more than one processor. While this has yet to be performed, it seems like a relatively easy task. The time accurate solution was marched in time with 0.01 sec time steps with 20 sub-iterations. FLUENT states that the ideal number of sub-iterations is between 10-20 depending on the size of the time step, so 20 was taken to ensure good results. Also, for the cylinder example, it was shown that only 10 sub-iterations, with a fairly large time step, is not sufficient to obtain a time accurate solution. The grid initially contained 13,179 cells, and had 13,602 cells after one period of oscillation. The increase in the number of grid points can be seen in the bottom right-hand picture of figure 11 where a trail of cells has been created due to the rotation of the

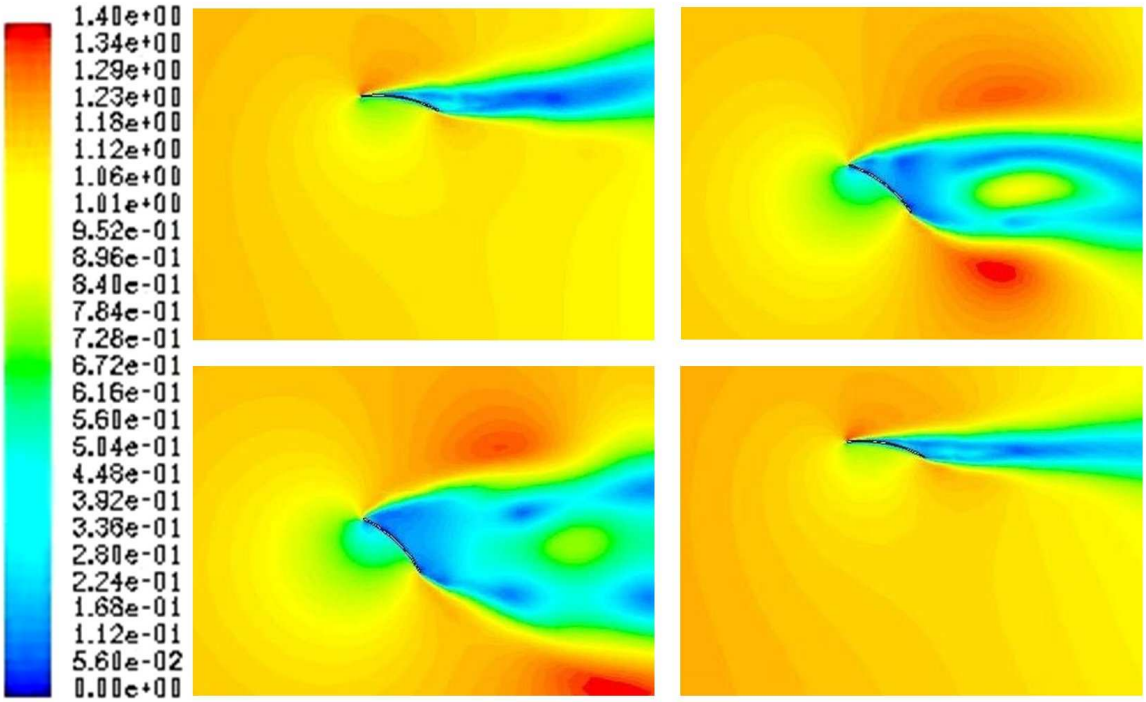


Fig. 12. Velocity magnitudes at different instances in the cycle of the flapping motion.

moving portion of the circular arc. Figure 12 shows velocity magnitude contour plots for the series of meshes shown in figure 11.

While the fluid solution coupled with the moving deforming mesh proved the capabilities of FLUENT's moving and deforming mesh, the quantitative nature was not satisfactory in matching any results from the literature.[16] However, this was to be expected because the results from the cited literature are from fully three-dimensional (3D) flow while this is only a two-dimensional calculation. Also, the customer suggested Reynolds number of 5000 did not match the literature Reynolds number. Later investigation into the solution found that different boundary conditions should be chosen. The goal of this exercise was to verify the remeshing techniques and run a flow solution to test FLUENT. Still with the data at hand,[16] a quick comparison reinforced the notion that no comparison should be made between the two.

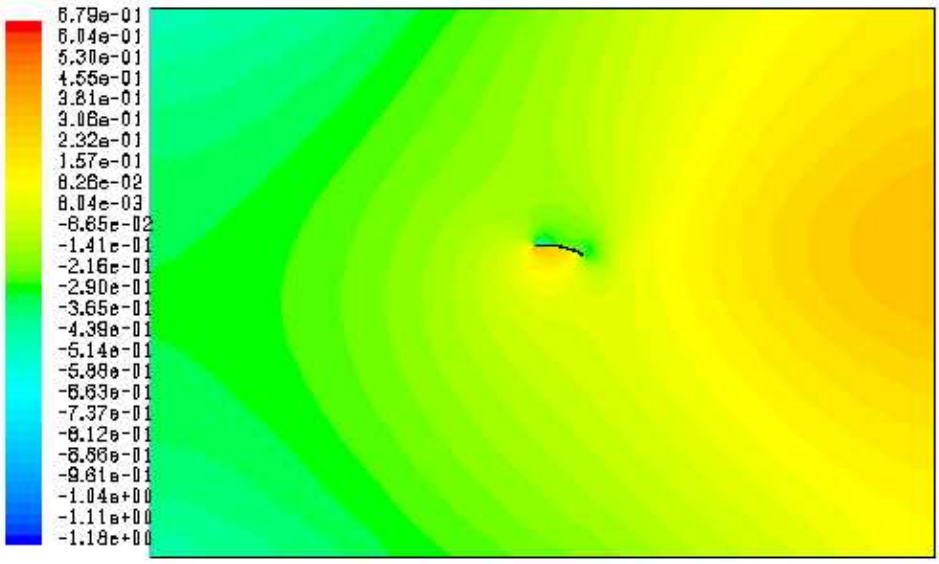


Fig. 13. Static pressure contour of entire domain showing pressure build-up at exit.

Two factors concerning grid quality should be considered in this simulation:

1. The boundaries were not moved far enough away from the arc such that they have a minimal effect on the aerodynamics. This can be seen in figure 13. The static pressure seems to be high at the outlet boundary . If the boundary conditions were truly exit boundary conditions there is no reason why there would be a buildup of static pressure in this region. This phenomenon did not exist for the circular cylinder flow solution with boundaries at 50 diameters away. Pressure outlet boundaries would probably be better with the grid which is used. That boundary condition allows the user to specify what the exit static pressure of the flowfield must be and thus help eliminate any pressure build up.

2. The grid resolution is probably not refined enough to capture the effect that the shed vorticies have on the fluid solution. This fact can be justified by realizing that in the initial stage of having a moving deforming mesh and running a fluid solution, only a minimal number of cells were used to test the method, rather

than focusing extensively on the fluid solution. It is believed that a successful method has been constructed to study fluid-structure interaction of rigid bodies in two-dimensions by using FLUENT. Of course this conclusion comes with the added caveat that says time and care must be taken to ensure grid quality at every time step of the moving boundary simulation.

E. Conclusions and Future Applications

The author was intrigued by the elastically mounted cylinder problem[17] and is very interested in investigating how a FLUENT UDF and moving deforming meshes can be used to numerically simulate this problem. The moving deforming mesh portion of the problem has already been shown to work. No additional tools are necessary to prepare the mesh for movement. Also, it has been shown that FLUENT solver is robust enough to capture unsteady phenomena. The next step would be writing a UDF that calculates the forces and moments. The author has found that there exits in FLUENT a macro by the name of COMPUTE_FORCE_AND_MOMENT. No documentation was found on this macro but it has been used to find the forces and moments acting on a body in the UDF "6DOF."[11] If the forces, both lift and drag, on the cylinder can be found then these forces can be used in the dynamical equations of motion for the elastically mounted cylinder. Writing a Runge-Kutta fourth-order solver to find the velocities and then using the define macro DEFINE_CG_MOTION to apply these new velocities to the movement of the cylinder would result in the numerical simulation of an elastically mounted cylinder. With the current tools FLUENT offers, this problem should be solved with minimal effort.

Fluid flow calculations of moving and deforming bodies has been shown to be possible and an algorithm of how to do this with FLUENT has been devised and is

shown. The implementations of solving moving bodies which are not deforming is powerful in and of itself. This can be used to study fluid systems where the loading causes small deflections. An excellent example would be the rotor-stator interactions of turbomachinery flows. Then after some additional validation, a fully aeroelastic analysis of rotor stator interaction might be solved. The next step of this research looks into how to model fluid flows where combustion is taking place. It is hoped that moving and deforming mesh together with combustion could be used to examine rotor-stator interaction of a combustion turbine or *in situ* reheat. More about this concept will be discussed in the following chapters.

CHAPTER V

VALIDATION OF COMBUSTION MODEL FOR *IN SITU* REHEAT WITH 3-D METHANE INJECTION VANE

In order to numerically model a combustive flow it is necessary to have a means of calculating how much of a certain species is used or generated during the combustion process. Not only how much, but also how fast these chemical reactions take place is of the utmost importance. This information about the destruction and creation of chemical species is often given by elementary reaction kinetics of combustion processes called reaction mechanisms. Ideally, these expressions for the rates of reactions should come from theory and thus satisfy all physical constraints of the flow process. Instead reaction mechanisms are typically empirical models, developed from physical experiments.

In order to use an empirical model to investigate *in situ* reheat, it is necessary to test the combustion mechanisms in a flow situation where experimental data is available with flow conditions similar to that of a jet turbine. This section presents the comparison between the experimental data and the numerical results for a single vane burner operating at conditions similar to an inlet guide vane of a typical power generation turbine. Because of experimental limitations, the total pressure upstream of the combustion probe is smaller than the total pressure upstream of the inlet guide vane of a typical power generation turbine. While not exact, the experimental setup of an inlet guide vane is very similar to turbine flow conditions, thus it is used in order to validate the combustion mechanism for flow conditions of this type.

This chapter will first discuss the experimental setup of the vane burner. It will then discuss the numerical setup and how a grid was generated to model this problem. The results will be compared with the physical experiment and some insights about

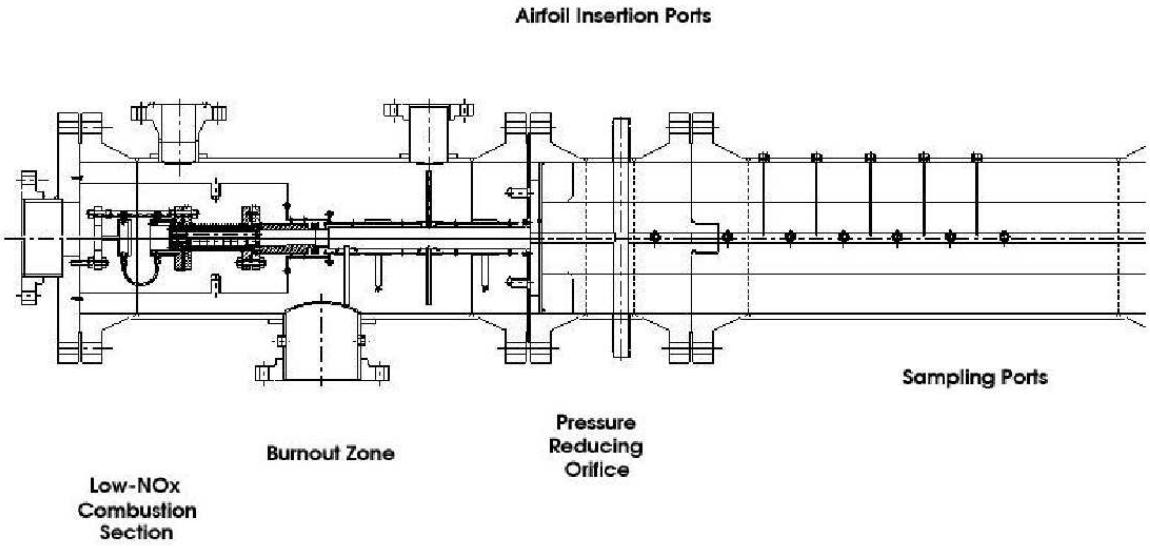


Fig. 14. Experimental setup for single-vane burner.

the specific reaction mechanisms used in this analysis will be discussed.

A. Experimental Setup

The experimental apparatus is shown in figure 14. The experimental tests were performed in the Siemens Westinghouse small-scale, full-pressure, combustion test facility. Preheated air and natural gas were delivered to a low-NOx burner section. Air temperature and fuel/air ratio were adjusted to give an exhaust gas stagnation temperature and composition corresponding to a selected location in a turbine cascade. The exhaust gas was then passed through a pressure reducing orifice to increase the Mach number in the injection and sampling sections to typical turbine levels. A back pressure control valve was used to set the sampling section pressure. Gases were sampled at various locations downstream of the injection point, and compositions determined using a gas chromatograph, with error limits of 5%. An idealized depiction of the single vane burner domain is shown in figure 15. The combustion vane was located inside a 1 *in* by 0.7 *in* rectangular tube. The geometry of the combustion vane

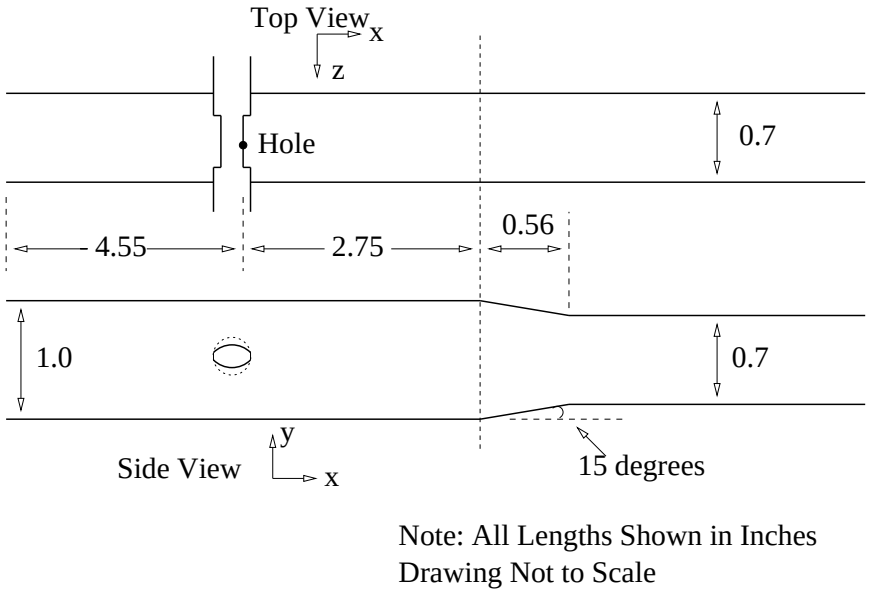


Fig. 15. Idealized experimental apparatus.

is shown in figure 16. Fuel was injected through a 0.026 *in* diameter hole located on the backside of the vane. Downstream of the injector vane, the tube section changes to a 0.7 *in* by 0.7 *in* square cross section. Temperature and gas composition were measured at several locations downstream of the fuel injector.

An already combusted fuel gas mixture enters the domain 4.55 *in* upstream of the vane and flows downstream. This mixture has a total pressure of 6.26 bar and a total temperature of $1507K$. The mass flow rate of the gas mixture entering the 1 inch by 0.7 inch rectangular cross section is 0.1345 kg/s. The composition of the gas mixture at the inlet of the tube is given in table III.

Experimental values for flow conditions at the injection hole are also given. The composition of the fuel entering through the hole is given in table IV. The temperature of the fuel is $289K$ and the mass flow rate is 0.416 kg/s.

The other flow condition which was given was that at the exit of the long narrowed (0.7in by 0.7in) domain, the static pressure is 4.6 bar. Everything else surrounding the domain are walls.

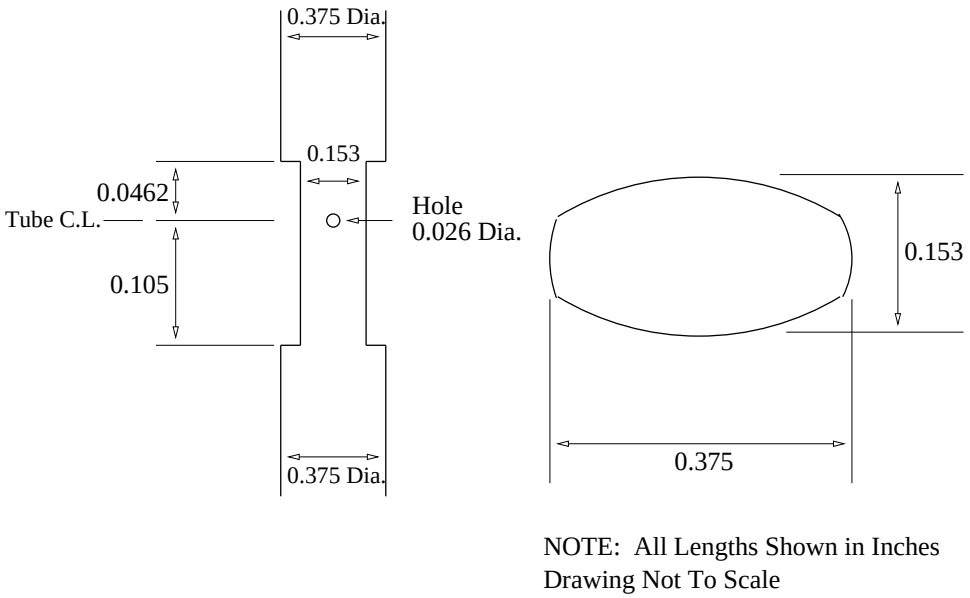


Fig. 16. Combustion probe geometry.

B. Numerical Boundary Conditions

Due to certain limitations, most notably in the combustion model chosen, some minor changes had to be made in order to simulate this experiment. The experiment had small amounts of ethane (C_2H_6) and propane (C_3H_8) which were injected through the vane. However, the combustion models which were used were relatively simplistic, which means they did not allow for transport equations for either ethane or propane. Because the volume percentages of these species were so low, the percentages of ethane and propane were simply lumped together with the volume percentage of methane. Therefore, the molar composition for the numerical model had 99% methane, 0.5% carbon dioxide and 0.5% nitrogen.

The flow parameters were calculated initially for the probe without fuel injection. This simulation provided the static pressure value at the fuel injection location. Consequently, it was assumed that the static pressure at the fuel injection hole was the same whether methane was injected or not. The fuel density was calculated knowing

Table III. Experimental inlet gas mixture molar composition percentage

Species	% Molar Composition
CO_2	4.84
H_2O	10.59
N_2	73.48
O_2	10.21
Ar	0.88

Table IV. Experimental fuel injection mixture molar composition percentage

Species	% Molar Composition
CH_4	96.1
C_2H_6	2.0
C_3H_8	0.9
CO_2	0.5
N_2	0.5

the pressure, temperature and fuel composition. After using this information to specify the boundary conditions of the problem, the injection velocity for the numerical simulation was checked against that of the experiment. The velocity of the simulation matched the velocity of the experimental test at the inlet of the domain as well as at the fuel injection hole. This is especially important because velocity boundary conditions are not specified anywhere in the problem.

The inlet boundary was treated as a pressure inlet boundary. This means that the total pressure, total temperature, direction of the flow, and species mass fractions

were specified at the inlet. Also, for turbulence quantities, the turbulence intensity and hydraulic diameter were set at the inlet. The turbulence intensity, I, is defined as the ratio of the root-mean-square of the velocity fluctuations, u', to the mean flow velocity, $\overline{U}$.[4] The turbulence intensity at the core of a fully-developed duct flow can be estimated with the following formula.

$$I \equiv u'/\overline{U} = 0.16(Re)^{-1/8} \tag{5.1}$$

This formula comes from empirical correlation for pipe flows, which resulted in a turbulence intensity close to 10%. One of many available turbulence length scales is the large eddy length scale. This quantity is related to the size of the largest eddies which are created in turbulent flows. In fully developed duct flows the size of the largest eddy is limited to the size of the inlet duct. For this simulation, the hydraulic diameter was set equal to the size of the inlet. The turbulence length scale, l, was then

$$l = 0.07L. \tag{5.2}$$

L is the hydraulic diameter, or the appropriate largest eddy length scale, and l is the turbulent length scale. The factor of 0.07 is based on the maximum value of the mixing length in fully-developed turbulent pipe flow.[4] This is only an approximation which is made within FLUENT. It is important to note that this approximation is not always valid. The standard $k-\epsilon$ model, a relatively simple, yet well established turbulence model, was used. However, only values for turbulence intensity and hydraulic diameter are specified. FLUENT converts these quantities in order to give boundary conditions for k, turbulent kinetic energy, and ϵ, turbulent dissipation rate, at the inlet. This is needed because the $k-\epsilon$ turbulence model gives transport equations

for the turbulent kinetic energy and turbulent dissipation rate. The turbulent kinetic energy is related to the turbulence intensity by

$$k = \frac{3}{2}(\overline{U}I)^2. \tag{5.3}$$

The turbulent dissipation rate is related to turbulence length scale by

$$\epsilon = \rho C_\mu \frac{k^2}{\mu}\left(\frac{\mu_t}{\mu}\right)^{-1}. \tag{5.4}$$

C_μ is the an empirical constant specified in the turbulence model, and μ_t/μ is the turbulent viscosity ratio. Therefore, boundary conditions for k and ϵ are dictated by specifying the turbulence intensity and hydraulic diameter. For more information on the implementation of the standard $k-\epsilon$ turbulence model used, please see the FLUENT users manual.[4]

After total pressure, total temperature, species mass fractions, turbulence intensity, hydraulic diameter, and the direction of the flow are specified, the static pressure and the inlet velocity magnitude are calculated within the program. The same type of boundary condition was used to model the hole on the vane where the fuel was injected. Table V shows the boundary conditions specified for the inlet and fuel injection vane.

At the exit of the domain, a pressure outlet boundary condition was applied. With this, the exit static pressure and flow direction are specified. Other boundary conditions are specified if back-flow occurs. However, for this problem, back-flow only occurred during the first few iterations.

The type of wall boundary conditions used for the simulation are no-slip adiabatic walls. Thus the velocity along the walls was zero and there was no heat transfer from the domain to the surroundings. The later of the two wall boundary conditions is

Table V. Input data for vane-burner

Parameter	Inlet	Injection
Total Pressure $[bar]$	6.26	7.95
Total Temperature $[K]$	1507	311
Turbulence Intensity $[\%]$	10	10
Hydraulic Diameter $[m]$	0.0254	0.00066
Mass Fraction		
CH_4	0.000	0.9778
O_2	0.1150	0.000
CO_2	0.0754	0.01355
CO	0.000	0.000
H_2O	0.06755	0.000
N_2	0.74205	0.00865

important to this problem. As will be shown later, the geometry of the tube is quite long, which allows heat to be lost through the wall boundaries of the vane burner. More about the effects of the adiabatic wall boundary assumption will be discussed in the results section.

C. Grid Generation

The creation of the grid was done using the grid generation software package that is available with FLUENT called GAMBIT. The geometry of the domain required certain important parameters be taken into account. The total single-vane burner had a length of 1.18 m. However, the height and width of the burner are only 2.54 cm

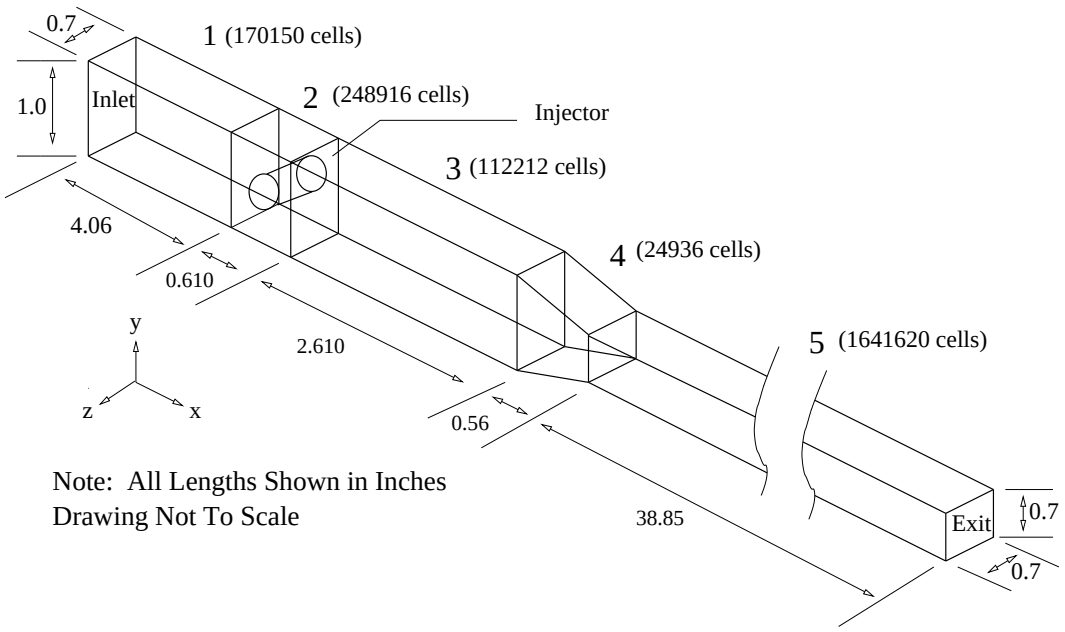


Fig. 17. Idealized illustration of numerical domain.

and 1.778 *cm* respectively at the inlet. The single vane burner is located about 10 *cm* downstream of the inlet. Another 10 *cm* downstream of the vane the height decreases until it reaches 1.778 *cm*. The 1.778 *cm* square cross section remains constant for about 90 *cm* downstream. It became necessary to make the computational domain this large because it was important to allow the 3D flow to develop before reaching the vane. The large domain is also needed because the numerical simulation would be compared with experimental measurements which had taken place as far as 83.6 *cm* downstream of the injector. Therefore it was a balancing act of how far the boundaries could be placed away from the region of interest and how computationally expensive the simulation would be.

The primary problem in generating the grid for this simulation was being able to allow enough cells in the cross section of the single vane burner and still have an appropriate number of cells in the axial direction of the burner without having an unfeasible number of total cells. Steps were taken to ensure that the grid was created

to have an ideal number of cells in the the cross section with minimal cell skewness in the axial direction. Figure 17 shows the entire domain of the single-vane burner. The entire geometry was broken into 5 domains. The second domain contains the fuel injector. Because this section contained complex 3D geometry, the vane equipped to inject fuel, it was meshed using tetrahedral elements. These elements are more capable of capturing the intricacies of the geometry. The remaining zones had elements which were composed of triangular prism cells. The *yz*-plane of each cut shown was created using triangular 2D elements which were extruded in the axial direction to create the rest of the domain. The last section, section 5, is an extremely long section. Figure 17, cuts this region to show the entire domain in a manageable fashion. The last region was important for a comparison between the numerical simulation and experimental measurements far downstream. Thus cell quality and quantity had to be maintained throughout this long section. A breakdown of the cell number and type in each section is shown in table VI.

Table VI. Numerical grid size information

Grid Section	# of Cells	Cell Type
1	170,150	Triangular Prism
2	248,916	Tetrahedral
3	112,212	Triangular Prism
4	24,936	Triangular Prism
5	1,641,620	Triangular Prism

No boundary layer cell clustering was performed. This would have doubled the number of cells in the domain because almost all of the domain is surrounded by wall boundaries. Instead, wall functions are used in regions near wall boundaries.[4]

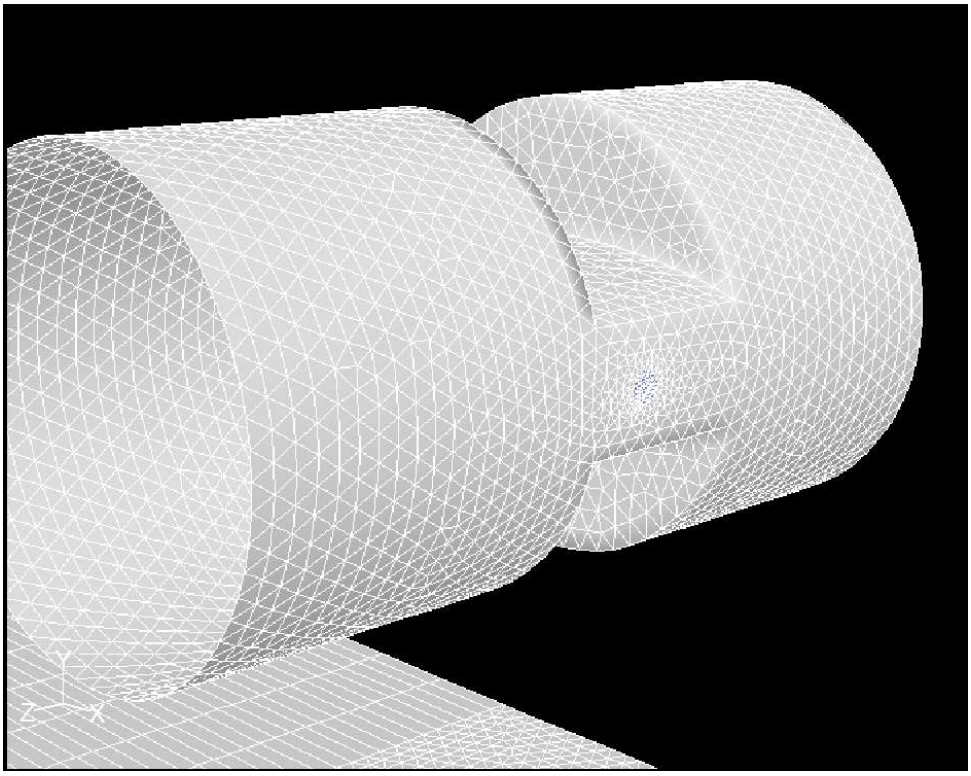


Fig. 18. Detail of fuel injector.

When using wall functions the viscosity-affected inner region of the boundary layer is not resolved. Instead, semi-empirical formulas are used to model the viscosity-affected region between the wall and the fully turbulent flow. Much is known about a turbulent boundary layer of a flat plate, so wall functions are fairly sophisticated formulas, even though they are not as ideal as boundary layer cell clustering. Wall functions substantially save computational resources because the wall regions do not need to be resolved. Since the grid was already computationally expensive at 2.2 million cells, wall functions were necessary to model the boundary layer regions.

The shape of the vane burner was defined by the intersection of two radii. The injection hole had a diameter of 0.66 *mm*. The injection hole was located at the center of the pipe, however, the shoulders of the vane were not equally-spaced with respect to the injection hole. A detailed figure of the computational grid of the single-vane

burner is shown in figures 16 and 18.

D. Combustion Model Used in Simulation

An important step in the simulation of a combustive flow is the selection of the chemistry model. For the research performed herein a two-step, global, finite rate combustion model is used for methane and combustion gases.[18] The same basic model was used for two simulations, however there was a slight difference between the two simulations which will be discussed with the formal introduction of the chemistry models in what follows.

1. Chemical Model A

The first model used is a two-step finite rate combustion model which uses the following chemical reactions

$$\begin{aligned} CH_4 + 1.5O_2 &\rightarrow CO + 2H_2O \\ CO + 0.5O_2 &\rightarrow CO_2. \end{aligned} \tag{5.5}$$

Therefore, the combustive simulation will model the species transport of six species. Five of the species come from the chemical equations in (5.5). The sixth species, nitrogen, is the species of greatest concentration throughout the domain. Also, as mentioned before, the species mass fraction for nitrogen is not calculated but is actually found after determining the species mass fractions of the other species. This is because of the requirement that the sum of the species mass fractions must be unity. The expression for the forward rate constant is computed using the Arrhenius expression and has the form

$$k_1 = A_1 \exp\left(E_1/R/T\right) \left[CH_4\right]^{-0.3} \left[O_2\right]^{1.3}, \\ A_1 = 2.8 \cdot 10^9 \ s^{-1}, E_1/R = 24360K. \tag{5.6}$$

Something important to note is the difference between the expression used to calculate the rate constant and pure Arrhenius law of equation (2.30). The different terms are $[CH_4]^{-0.3}$ and $[O_2]^{1.3}$. These terms are the concentration of the specific species named with a corresponding concentration exponent. They are added because of the empiricism of the chemical model. The rate constant for the carbon monoxide oxidation has the following expression

$$k_2 = A_2 \exp\left(E_2/R/T\right) \left[CO\right] \left[O_2\right]^{0.25} \left[H_2O\right]^{0.5} \\ A_2 = 2.249 \cdot 10^{12} \left(m^3/Kmol\right)^{0.75} s^{-1}, E_2/R = 20130K. \tag{5.7}$$

In this model the temperature exponent of the Arrhenius expression has been set to zero. The remaining terms are either already known or have been empirically derived. Specifically, the pre-exponential A and the concentration exponents are empirically derived quantities.

2. Chemical Model B

The second chemical model is almost exactly like model A, but with the addition of a reversible reaction for carbon monoxide. Therefore the chemical expression has the following form

$$CH_4 + 1.5O_2 \rightarrow CO + 2H_2O \\ CO + 0.5O_2 \leftrightarrows CO_2. \tag{5.8}$$

In order to implement this model within FLUENT, the user is allowed more freedom by defining a third reaction instead of just a reversible reaction. More freedom meaning that the user is allowed to specify different concentration exponents,

pre-exponential factor and activation energy. Because this model is empirically derived, different values for the pre-exponential factor and concentration exponents are necessary to describe the reversible reaction. The first two rate constants are calculated exactly the same as in model A. The third rate constant, or the rate constant describing the reversible reaction for carbon monoxide oxidation, has the following expression:

$$\begin{gathered} k_3 = A_3 \exp\left(E_3/R/T\right)\left[CO_2\right]^1 \\ A_3 = 5.0 \cdot 10^8 \left(m^3/Kmol\right)^{0.75} s^{-1}, E_3/R = 20130K. \end{gathered} \tag{5.9}$$

Detailed chemical models show that there exists a burned gas equilibrium ratio for [CO]/[CO_2]. Adding the reversible reaction allows the model to better reproduce the pressure dependence of this [CO]/[CO_2] equilibrium as well as give a better representation of the heat of reaction.[18] Thus, it is assumed that adding the reversible reaction will help the simulation capture true physical phenomena. In the next section, results will be shown for both combustion models and a comparison with experimental data will be made.

E. Results

In the beginning, a two-dimensional simulation of the single vane burner was performed. The two-dimensional approximation was obtained by taking a cut at the $z = 0$ plane of the three-dimensional injector and performing the combustion simulation on only this plane. Three fuel injection cases were considered in the 2-D numerical simulation: (1) the length of the fuel injector was equal to the diameter of the hole at the $z = 0$ plane, that is, 0.66 mm, (2) the size of the fuel injection was set by examining the ratio of areas of the three-dimensional problem. Specifically, the area of the injector hole divided by the area of the inlet of the 3D experiment should be proportional to the height of the fuel injector divided by the height of the inlet in the

two-dimensional simulation, and (3) the length of the injection hole was a geometric average of the lengths used in cases (1) and (2).

The results for the 2D simulation were not encouraging. None of the values which were experimentally measured seemed to match the 2D simulations. The single-vane burner is a long rectangular box and the details of the vane injector show that the injector is only symmetric about the $y = 0$ plane. Thus, it became apparent that the flow is fully three-dimensional and requires a full three-dimensional simulation. The remaining results will focus on the 3D simulations.

Two different three-dimensional simulations were performed using each of the two different chemical models described previously. The results from each simulation will be discussed.

1. Results of 3D Injector Simulation with Chemical Model A

The first set of results will show a steady state simulation of the flow through the numerical domain. Ideally, an unsteady simulation would be best, as the geometry of the vane suggests some unsteadiness might exist. However, due to computational expenses, a steady state computation was the only option feasible. The computational expenses came from being forced to solve transport equations for turbulence kinetic energy, turbulence dissipation rate, and species balance equations for five different species, in addition to solving mass, momentum and energy equations for a total of 12 transport equations over 2.2 million cells. The grid and data files were quite large, about 415 megabytes in size, and the simulation takes 1.5 gigabytes of RAM memory to run. All computation was performed at the Texas A&M University Supercomputing Facility. A steady state calculation took an IBM Regatta p690 supercomputer approximately 195 hours wall clock time while running in parallel on four processors. This is equivalent to about 8 days of constant running using four processors.

Experimental species data was given at two axial locations within the burner. Mole fraction percentages were given for species of methane and carbon monoxide at axial locations of 0.311 m and 0.654 m downstream from the vane. At 0.836 m downstream of the vane temperature data was taken. The experimental results are given in tables VII-IX.

Table VII. Species mole fraction % at 0.311 m downstream using chemical model A

Parameter	Experimental	Centerline	Area-weighted average	Mass-weighted average
CH_4	0.35	3.52e-05	1.11e-05	1.20e-05
CO	0.16	2.96e-04	9.70e-05	1.05e-04
CO_2	N.A.	05.34	05.37	05.36
O_2	N.A.	08.97	08.91	08.92
H_2O	N.A.	11.58	11.63	11.62

Table VIII. Species mole fraction % at 0.654 m downstream using chemical model A

Parameter	Experimental	Centerline	Area-weighted average	Mass-weighted average
CH_4	0.08	7.45e-08	1.28e-08	1.47e-08
CO	0.27	9.89e-07	1.75e-07	2.01e-07
CO_2	N.A.	05.31	05.37	05.36
O_2	N.A.	09.04	08.91	08.92
H_2O	N.A.	11.52	11.63	11.62

Table IX. Temperature values at axial locations using chemical model A. Experimental value at 0.836 m is 1478K

	0.311 m	0.654 m	0.836 m
Centerline			
Temperature K	1554	1537	1536
Area-weighted average			
Temperature K	1570	1564	1561
Mass-weighted average			
Temperature K	1569	1564	1561

In addition to the experimental results, numerical results are presented in three different manners. The first is the centerline value or the value at the centroid of the cross section of the specific x = constant plane. The other two values given are averages over the entire x = constant plane. Tables VII and VIII show mole fractions of all the species calculated in the simulation for each of the three manners used to describe the numerical results. Table IX shows the temperature values of the numerical simulation at each axial location. The experiment measures 1478K at an axial location of 0.836 m.

Comparing the numerical simulation with the physical experiment many differences can be seen. The species mole fractions for methane and carbon monoxide measured in the experiment are not matched at either axial location. The physical experiment shows much higher mole fractions for methane and carbon monoxide. Therefore, the numerical reaction model without the reversible reaction was not able to match species concentrations downstream. This is most likely due to the fact that the chemical model dictates the rate at which methane is destroyed. In the simulation

methane is completely consumed well before it reaches the contracted region of the domain. Therefore, almost no methane is found at 0.311 and 0.654 m downstream.

Figures 19-21 each show a series of contour plots. Each figure corresponds to a different quantity being plotted. Figure 19 shows a series of temperature contour plots inside the domain. The domain has been divided into certain x and $z =$ constant planes and an isometric view is given which shows the $z = 0$ plane with various $x =$ constant planes. The plots show the locations within the domain where the highest temperatures occur. It also clearly shows where the cold methane is being injected. Figure 20 shows a series of methane mass fraction contour plots similar to the temperature contour plots. Figure 21 shows the carbon monoxide mass fraction contour plots. These specific species are shown for two reasons. First, they are the species measured by the experiment, and second, because the chemical model used for this simulation deals with methane and carbon monoxide oxidation expressions. Therefore showing these contour plots will give some indication of how well the reaction model is simulated.

As a collection, the figures are also important because they describe the numerical solution in a different manner, which can also be contrasted to the experimental results. The methane being injected into the domain seems to burn quickly in the simulation as shown in figure 20. Only 35 mm after the methane is injected it is almost entirely consumed. This can also be seen by looking at figure 19. Where the methane concentration is high, the temperature is low in the domain. As the methane travels downstream it burns and releases heat energy, which in turn heats up the regions shown in figure 19 shown by the dark red spots. Also, further energy is being given off by the carbon monoxide oxidation. No carbon monoxide enters into the domain from the boundaries, rather all the carbon monoxide is generated by the methane oxidation. The regions where the carbon monoxide levels are the highest

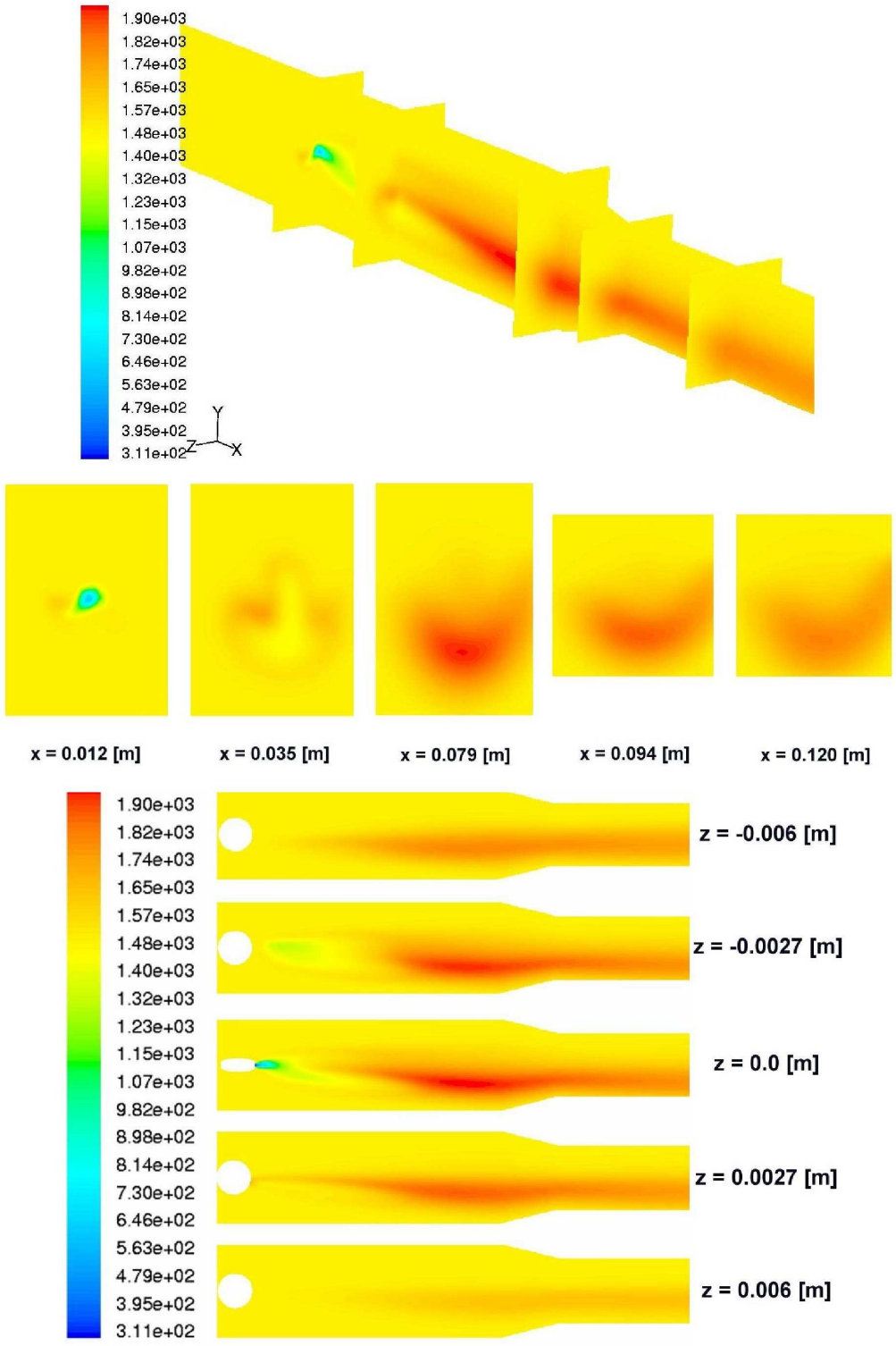


Fig. 19. Series of temperature contour plots using combustion model A.

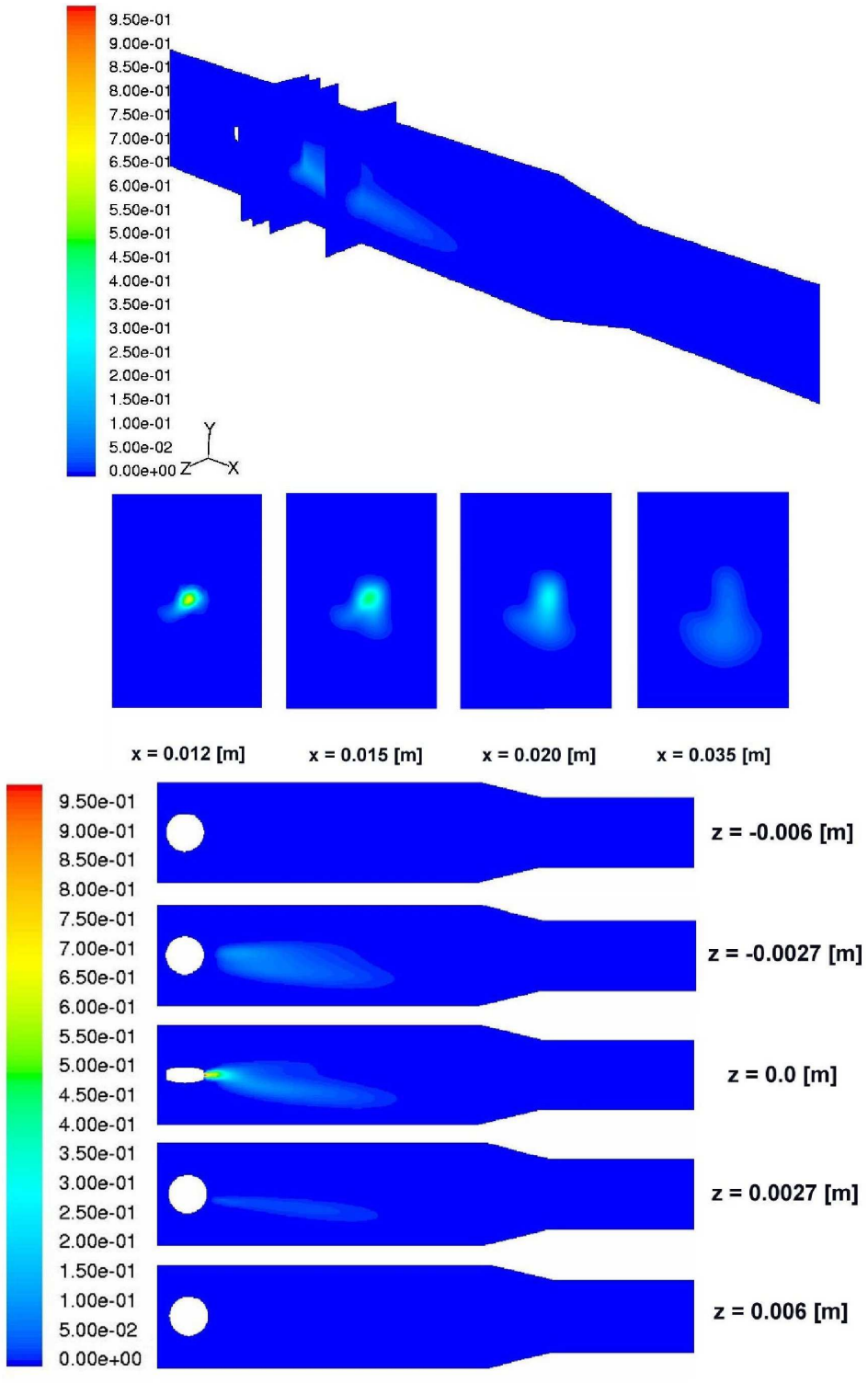


Fig. 20. Series of methane mass fraction contour plots using combustion model A.

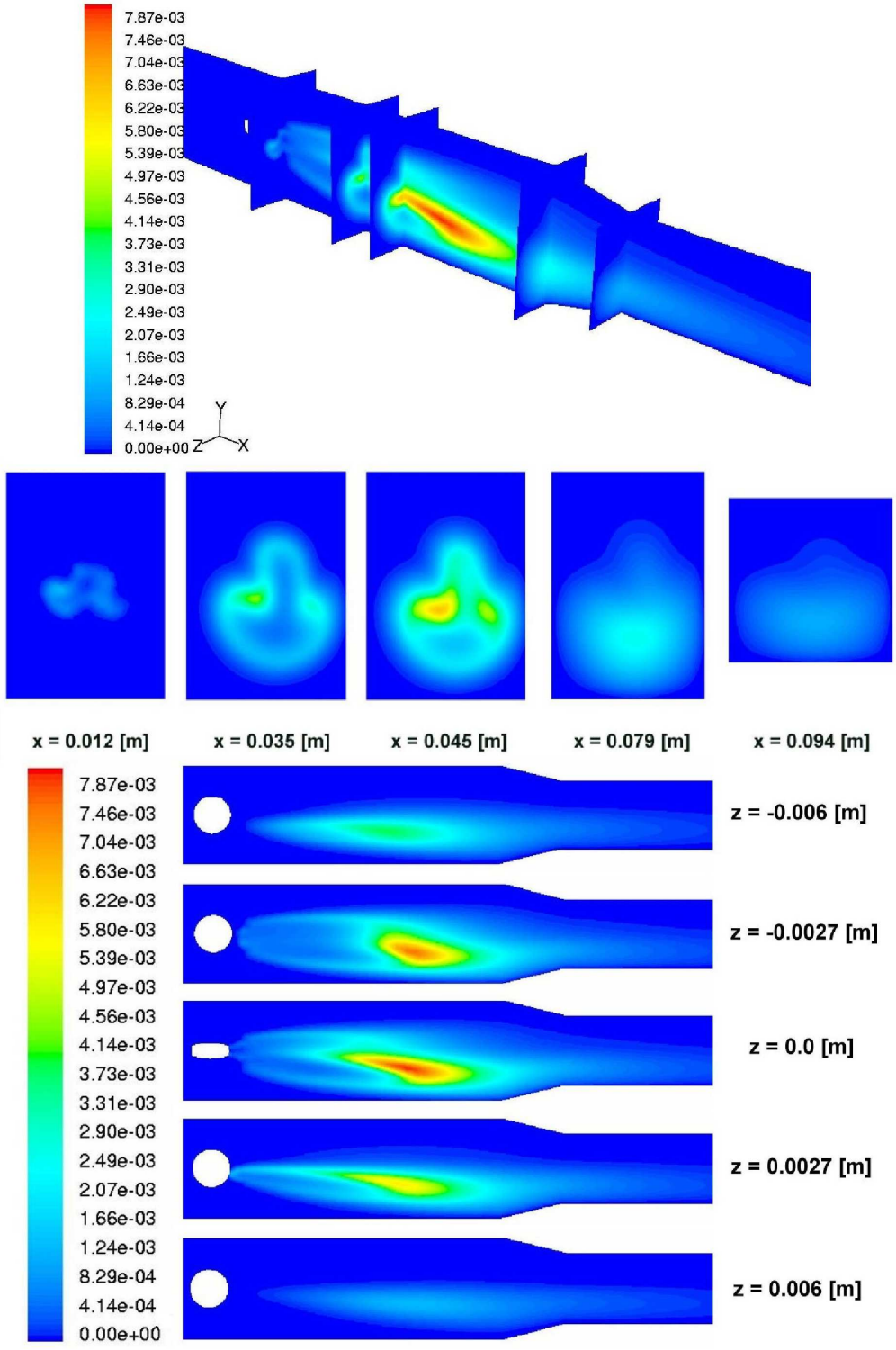


Fig. 21. Series of carbon monoxide mass fraction contour plots using combustion model A.

are where methane is burning and creating the most carbon monoxide. These points are at the same locations as the high temperature values.

The chemical model used in the simulation burns the methane much faster than the experiment data suggests. This is clearly evident by looking at tables VII and VIII. However, the simulation seems to over-predict the heat release given from the chemical reaction. This can be seen by looking at table IX. The temperature measured by experiment is $1478K$ while the closest numerical result measures $1536K$ at the centerline. The agreement between the temperatures is only about a 4% difference. Part of the difference can be attributed to the modeling of the walls as adiabatic. It is entirely likely that heat was lost through the walls during the experiment. The primary focus of these elementary reaction models is to match temperature or flame speed for combustion simulations, and when detailed chemical concentrations are needed, more equations are generally modeled. The essential element to study *in situ* reheat is to capture the temperature variation correctly, and while it would be nice to get detailed chemistry data, it is not imperative at this point in the research.

2. Results of 3D Injector with Chemical Model B

The focus of this research is to evaluate a reaction mechanism in a flow situation similar to what might exist in a turbine in order to study *in situ* reheat in a turbine. Since little is known about *in situ* reheat, it is of primary interest to this research to help develop an inexpensive reaction model for use in a turbine burner. This is because parametric analysis of fuel location, temperature, pressure, and flow rate need to be investigated to provide an idea of the positive and negative effects when an actual turbine burner is tested. To accomplish this, the chemistry model needs to be fairly accurate, especially with the temperature measurements, while at the same time being computationally inexpensive so that many different simulations can be

run. Therefore, any new model must still be relatively inexpensive when compared to the initial model, chemical model A. The reaction model used in this simulation adds the reversible reaction of carbon monoxide oxidation. As mentioned before, adding the reversible reaction allows the model to better reproduce the pressure dependence of this $[CO]/[CO_2]$ equilibrium as well as give a better representation of the heat of reaction.[18] This model should give better results and be almost identical in terms of computational expense because a new transport equation is not needed. In fact, this is exactly what was found. Results show better agreement with the experimental data when the reversible reaction, chemical model B, is included.

The results will be shown in exactly the same manner as was done with the previous combustion model. The only difference between the two simulations is the modeling of the reversible reaction of carbon monoxide oxidation. Tables X-XI show the species concentrations at 0.311 and 0.654 m downstream from the vane respectively. This simulation matches the carbon monoxide levels within the same order of magnitude as the experiment, particularly at 0.654 m downstream. The area-weighted average shows only a 9% difference. This is considerably better than the previous reaction mechanism, chemical model A.

Continuing the comparison, it is seen that the temperature levels also show better agreement with the experimental data. This comparison is seen in table XII. Instead of a 4% temperature difference, the simulation with the reversible reaction shows as small as a 1.2% difference. Unfortunately, the methane mole fraction comparison did not improve. The numerical simulation basically showed no methane at the two axial locations of interest just as it had using the previous reaction mechanism. This makes sense because the expression for the methane oxidation is exactly the same in both models.

Table X. Species mole fraction % at 0.311 m downstream using chemical model B

Parameter	Experimental	Centerline	Area-weighted average	Mass-weighted average
CH_4	0.35	3.52e-05	9.62e-06	1.05e-05
CO	0.16	0.692	0.368	0.390
CO_2	N.A.	04.71	04.98	04.97
O_2	N.A.	09.10	09.09	09.09
H_2O	N.A.	11.71	11.60	11.61

Table XI. Species mole fraction % at 0.654 m downstream using chemical model B

Parameter	Experimental	Centerline	Area-weighted average	Mass-weighted average
CH_4	0.08	4.84e-08	7.99e-09	9.18e-09
CO	0.27	0.616	0.295	0.316
CO_2	N.A.	04.72	05.06	05.04
O_2	N.A.	09.19	09.05	09.06
H_2O	N.A.	11.60	11.61	11.61

Table XII. Temperature values at axial locations using chemical model B. Experimental value at 0.836 m is 1478K

	0.311 m	0.654 m	0.836 m
Centerline			
Temperature K	1514	1496	1496
Area-weighted average			
Temperature K	1542	1543	1542
Mass-weighted average			
Temperature K	1542	1541	1540

Further comparison of the two reaction models is made by looking at figures 22-24, which are similar to figures 19-21, also found at the end of the chapter. The first notable difference between the results using different combustion models comes from looking at the temperature contour plot, figure 22. This particular series of contour plots shows that the region of hottest gases is more spread out than what is seen in figure 19. This can be seen by looking at both the $x =$ constant and $z =$ constant planes. Also, the flow does not get quite as hot compared to the results in the previous solution. This is because some of the energy that was going into heat is now being used to create the reversible reaction of carbon monoxide oxidation. In figure 23 the series of methane mass fraction contour plots look very similar to the previous contour plots. Neither shows much methane past 35 mm downstream. The major difference can be seen looking at the $z = 0$ plane. Instead of the methane traveling more toward the lower portion of the burner, figure 20, it appears to spread out more and go in all directions as it travels downstream, figure 23.

The last comparison is seen when looking at the carbon monoxide mass fraction

contour plots, figure 24. This series of contour plots varies greatly from figure 21. First of all, the scale had to be changed. This was because much higher levels of carbon monoxide are found in the simulation using the reversible reactions. Instead of having a mass fraction percentage maximum of 0.787, the maximum with the second reaction model is 1.5. Another important difference is that a small amount of carbon monoxide exists upstream of the vane injector. This is generated from the reversible reaction. The last major difference is the region where the highest concentration of carbon monoxide exits. Previously carbon monoxide reached its highest levels further upstream. In figure 24 relatively high levels of carbon monoxide exist even as the burner domain is contracting. The second chemical model clearly shows better agreement with the experimental data. Instead of having zero carbon monoxide mass fraction percentage at 0.311 m and 0.654 m downstream, the mass fractions match within an order of magnitude and the average values match as close as 9% difference between experiment and numerical simulation.

F. Conclusions and Recommendations

The major objective of this research was to test a chemical reaction model in a flow situation similar to what would exist in a turbine, where high temperature, high pressure gas with species concentrations similar to what exits a combustor is injected into a vane burner. The vane is equipped with an injector which injects primarily methane gas. The gases then combust as they travel downstream. It has been found that the first reaction model matched experimental temperature data by as close as a 4% difference at 0.836 m downstream of the vane. When utilizing a reversible reaction the results compared even better with temperature and carbon monoxide levels within 1.2% and 9% of experimental values, respectively. Because the

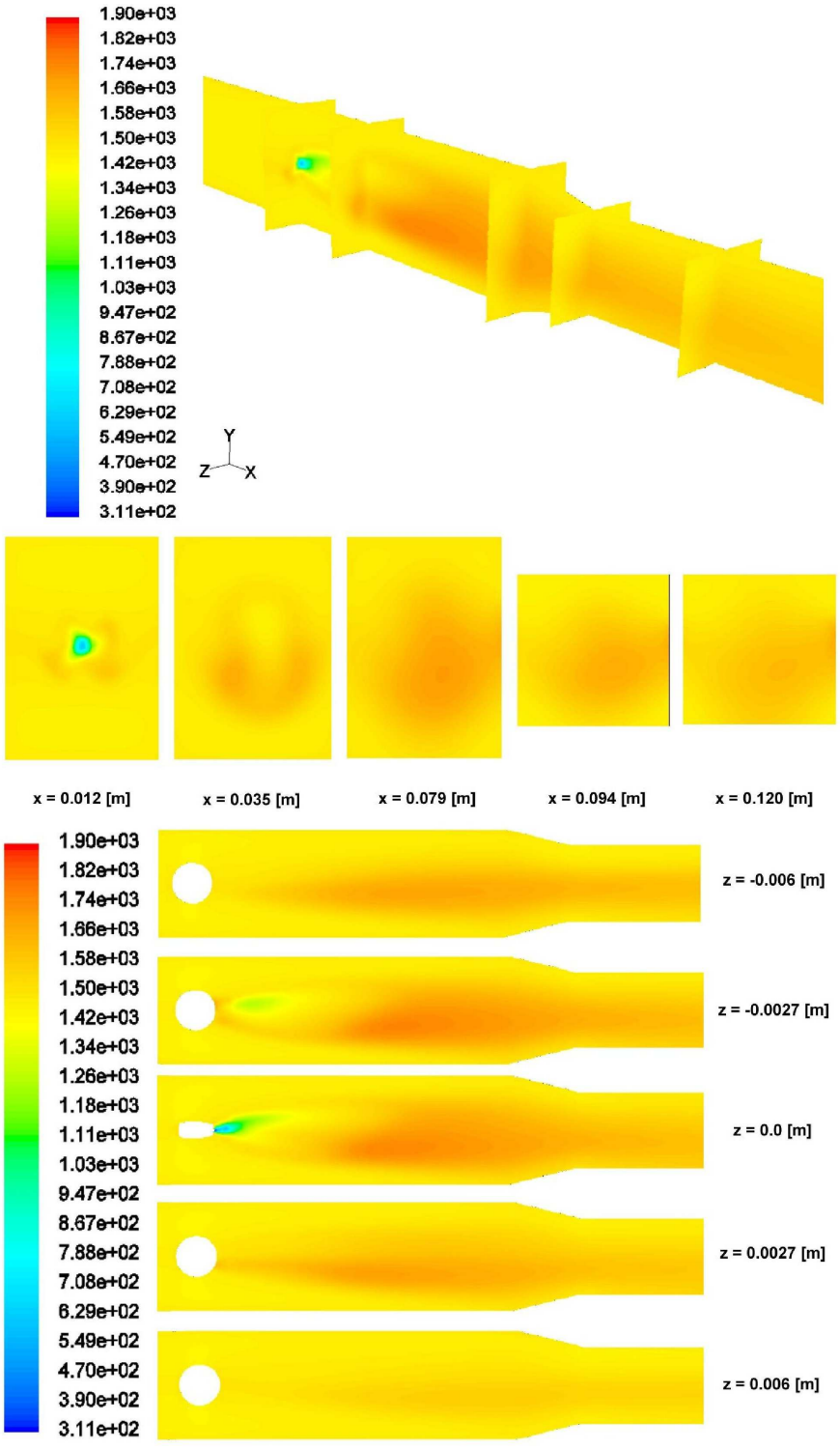


Fig. 22. Series of temperature contour plots with reversible reaction defined.

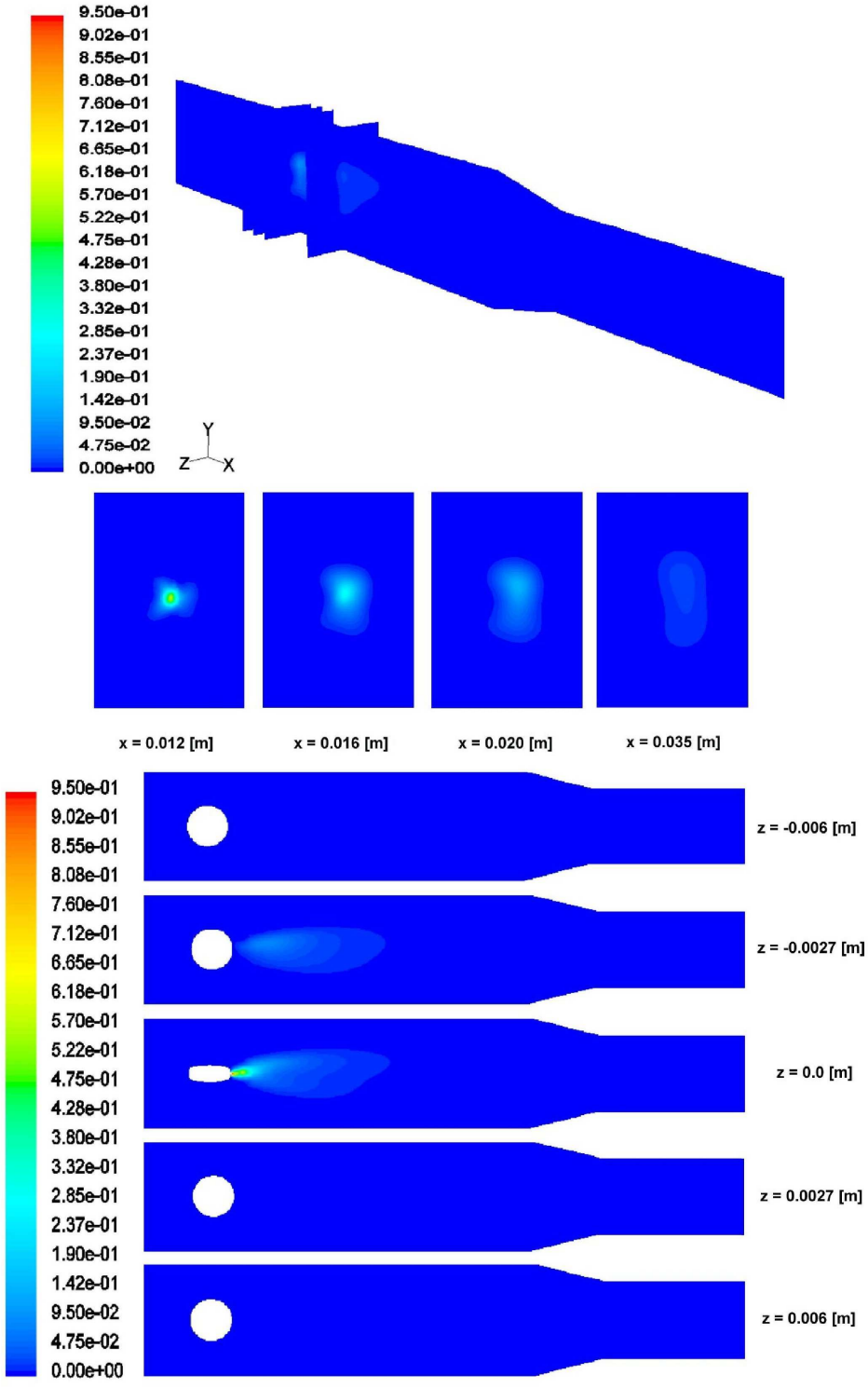


Fig. 23. Series of methane mass fraction contour plots with reversible reaction defined.

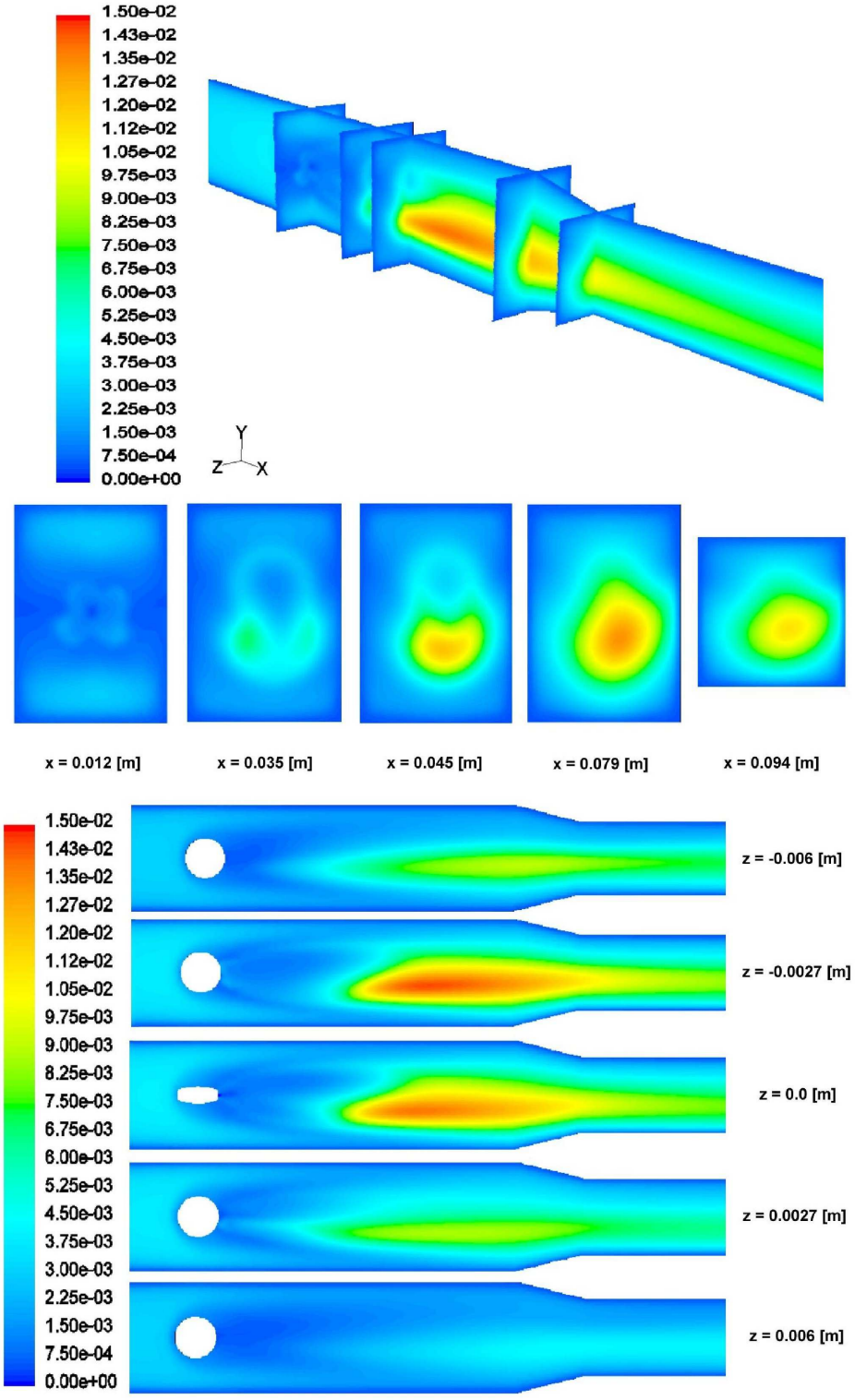


Fig. 24. Series of carbon monoxide mass fraction contour plots with reversible reaction defined.

reversible reaction compares much better and is only minimally more expensive, the second reaction model is recommended to use to investigate *in situ* reheat.

The results found from this research were applied to investigate *in situ* reheat using the CoRSI code to numerically model a turbine-combustor. A four-stage turbine-combustor was modeled by injecting methane into the trailing edge of the first stator. A temperature contour plot, figure 25, shows a snapshot of the unsteady combustive flow simulation performed.[19] This reference also includes a discussion of the CoRSI combustive flow software, developed at Texas A&M University. Additional information regarding the CoRSI code can be found by examining the following references. [20,21,22]

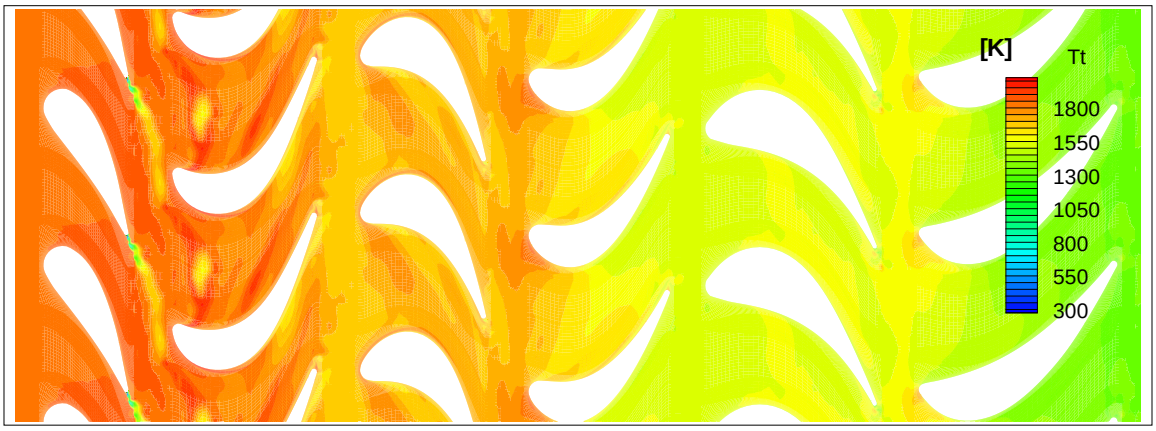


Fig. 25. Temperature contour plot of turbine *in situ* reheat simulation.

The numerical simulation predicted airfoil temperature variation and unsteady blade loading in the four-stage turbine combustor.[19] Also, *in situ* reheat was shown to increase the power generated by the turbine anywhere from 2.8% to 5.1% depending on the parameters of the fuel injection.[19]

Despite the findings of the current research, much more work is needed. A fully three-dimensional turbine model should be created to take into account the radial variation effects of *in situ* reheat. Also, the combustion model can be improved by

replacing the existing two-step combustion model by three-step, five-step or even sixteen-step combustion model. [23,24,25,26] However, any new model should be tested against the 3D single vane burner experiment presented here before it is considered to model a turbine-combustor.

Because the experimental data for the single vane burner is somewhat limited, it is desired to further test any combustion model with another experimental test. This was the approach taken in the chapter that follows. A simple two-step reaction mechanism is used to numerically simulate a laminar methane/air flame from Sandia National Laboratories. This reduces many of the complications found within a turbulent flow and allows the researcher to focus more on the combustion model being used. Also, the equipment available at Sandia's Combustion facility provides extremely accurate measurements. Further testing of the reaction mechanism against another experiment will provide a better indication of the limitations of the model. If a combustion model can show agreement with both experiments when it is implemented into a combustion simulation, it will be ready to be included in a turbine-combustor simulation.

CHAPTER VI

NUMERICAL INVESTIGATION OF A LAMINAR FLAME

The three-dimensional (3D) methane injector simulation discussed in the previous chapter validated the two reaction combustion mechanism for methane gas combustion. However, in order to look more specifically at how accurate the mechanism is, it was decided to verify the mechanism with a more rigorous experimental setup of a simpler flow. The 3D combustion vane performed earlier was a fully turbulent flow. Experimental measurements were only given at few points within the domain.

In order to study the effectiveness of the reaction mechanism only, the flow should be simpler. Thus, it became necessary to acquire experimental combustion data on a laminar methane/air flame.[27] This simplification removes the complications added by requiring a turbulence model and allows closer examination of the chemical model being used.

This chapter will begin by looking into the experimental setup of the problem. The next section will examine the numerical approximations and boundary conditions needed to simulate the experimental problem. Some discussion will be devoted to grid generation and the reaction mechanism used. A new method of analysis of combustion will be presented, which examines how well the simulation satisfies the second law of thermodynamics. To the knowledge of the author, up until now, no examination has been performed which asks whether or not the second law of thermodynamics is being satisfied in a combustion simulation. This question is particularly important because the use of empirical models does not guarantee the second law of thermodynamics is always satisfied. The entropy inequality used to perform this check as well as the numerical implementation of the inequality are given. Next, a comparison between the experimental measurements and the numerical simulation is shown. This leads to

an investigation of the numerical simulations ability to satisfy the entropy inequality. Last, a discussion of future work is presented.

A. Experimental Setup

This section will describe some details about the experimental setup of the problem. More specifically, it will describe the experimental parameters which were used in the numerical simulation that follows.

Sandia National Laboratories has a series of methane flame experiments that many studies have used as their experimental database for methane air combustion analysis. Different flames have different combinations of velocities and fuel mixtures. Each different Sandia flame is designated by letter. Flame A is a laminar flame methane/air combustion case which is the focus of the research presented here. The data set consists of temperature and mass fractions of all major species measured along the radius at three axial positions in the flame. The flame is basically axisymmetric as all experimental data are given with values starting at the center of the tube and traveling radially outward. Figure 26 shows the experimental setup used in the Sandia Turbulent Diffusion Flame Facility.

The whole experiment is open to ambient air. The flame is attached to the end of a long tube, with inner diameter of 7.72 *mm* and outer diameter of 9.525 *mm*. Figure 27 shows a simplified portrayal of the physical setup, as well as some of the dimensions of interest. A simple tube is used to avoid the complications of a flame pilot configuration. The length of the tube ensures a parabolic profile at the exit. A premixture volumetric flow rate of 6 *liters/min* of air and 2 *liters/min* of methane flows through the tube, resulting in an equivalence ratio of 3.17. The equivalence ratio is the actual fuel-oxidant mass ratio divided by the stoichiometric fuel-oxidant

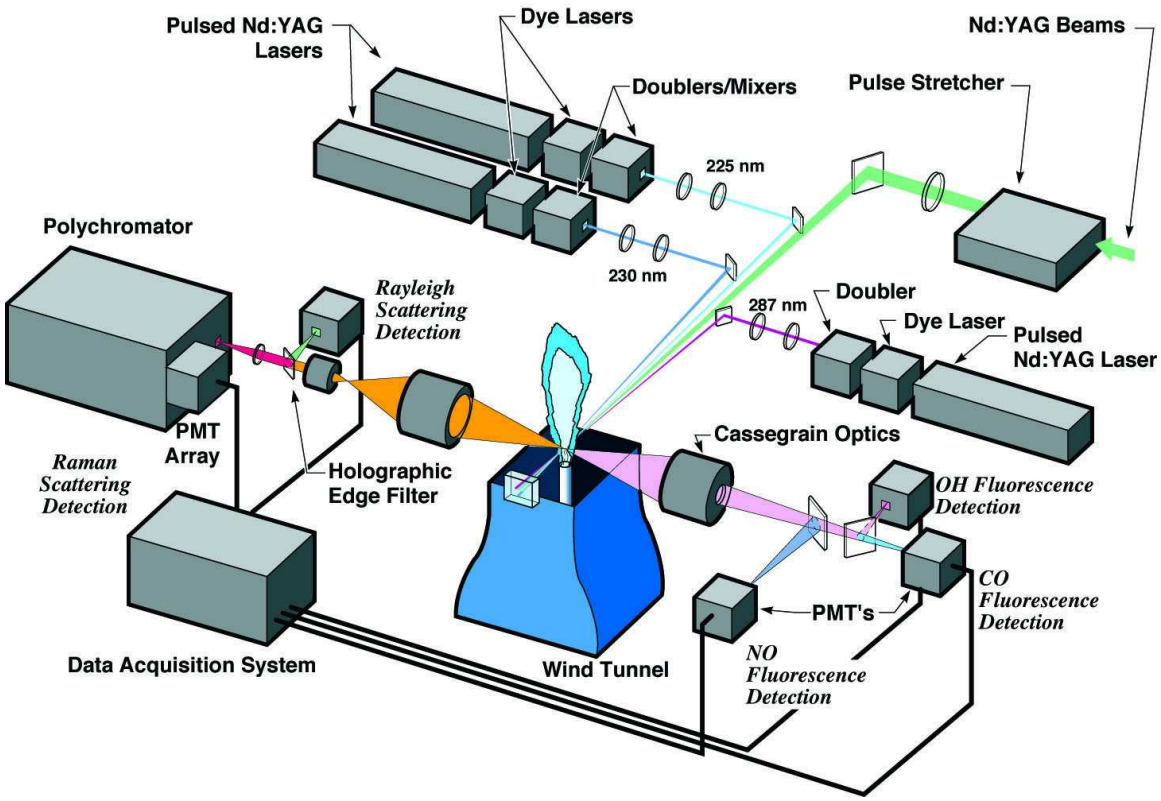


Fig. 26. Experimental setup of Sandia combustion flame facility.
Printed with permission of Robert S. Barlow, Sandia National Laboratories

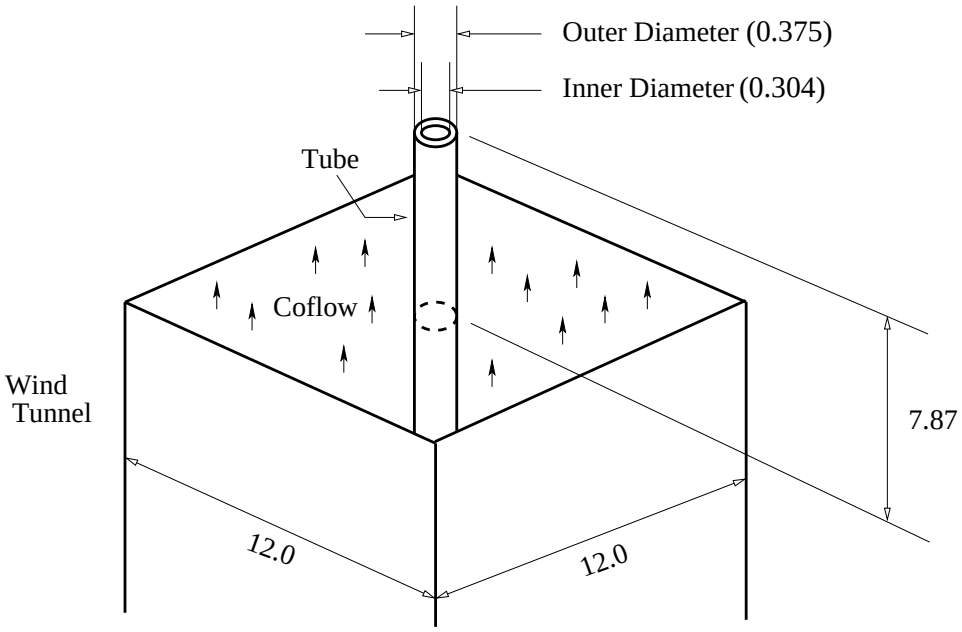


Fig. 27. Detail setup of laminar combustion flame.

ratio.[6] An equivalence ratio of 3.17 denotes fuel-rich conditions. There is a large vertical square wind tunnel section, 12 *in* x 12 *in*, surrounding the tube and creating a uniform co-flow of air traveling at 0.4 *m/s*. The square wind tunnel ends upstream of the flame, but the co-flow, or 0.4 *m/s* air generated by the vertical wind tunnel continues to move downstream and surround the flame.

Raman/Rayleigh and Laser-Induced Fluorescences of CO, OH, and NO were measured in single-point fashion. The resolution of the data is 500 microns. The flame was scanned at three axial positions as shown in figure 28 by the horizontal lines. The three axial locations were 25, 50, and 100 *mm* downstream. The scans consisted of 500 micron steps and an axis of symmetry was inferred from the data.

Both the co-flow and premixture into the domain are initially at ambient atmospheric pressure. The co-flow temperature is approximately 27 °C. The whole system is isobaric. The initial temperature of the premixed fuel is 27 °C also. This temper-

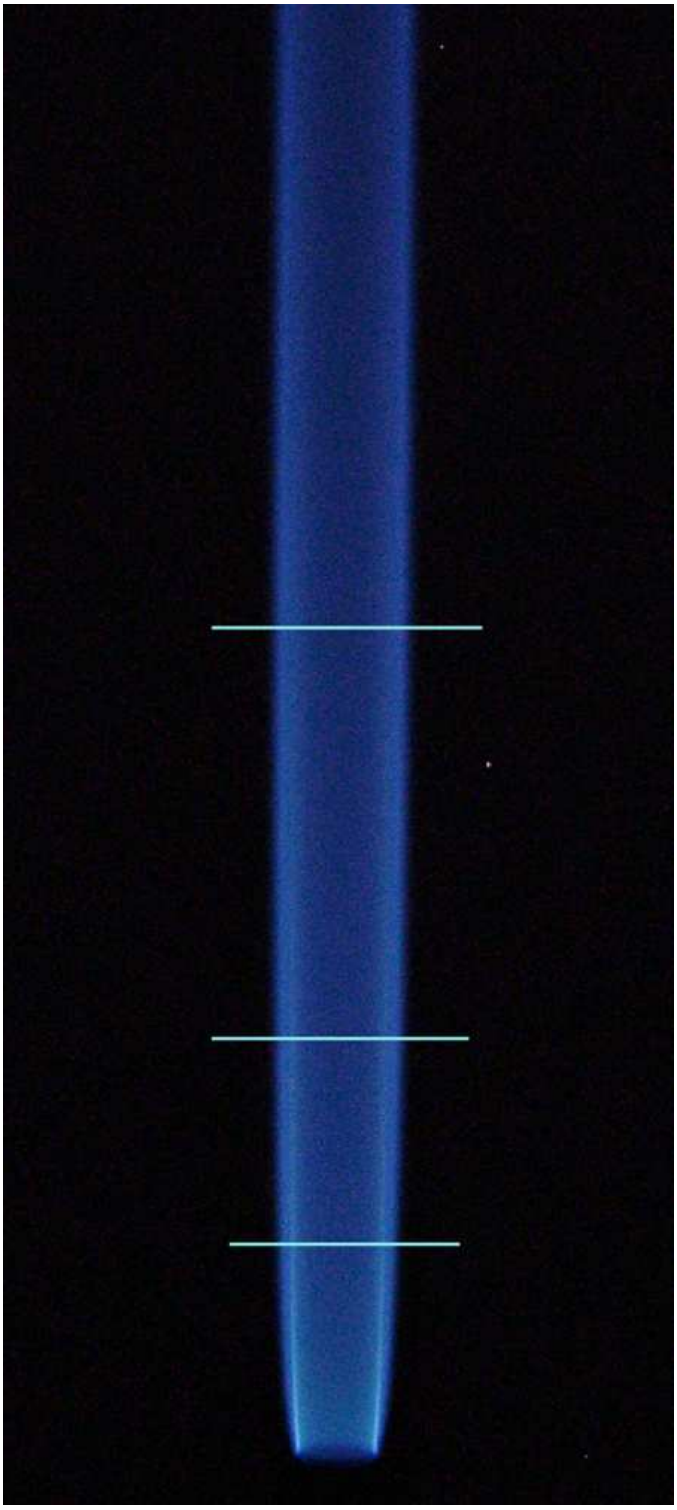


Fig. 28. Photograph of actual laminar flame.
Printed with permission of Adonios N. Karpetis, Sandia National Laboratories

ature applies when the premixed fuel is inside the tank and far from the exit of the tube. The actual temperature of the premixture is unknown when it exits the tube.

In the most strict meaning of the expression, there are no attached flames. There is always a small amount of air that comes into the flame in and around the tube. However, the amount of air is small when compared with large amounts that premix in a lifted flame case. In this experiment the flame is considered an attached flame.

B. Combustion Model

The type of combustion model used for this simulation is very similar to the combustion model B used in the 3D single vane burner. Both models are two step finite rate combustion models, consisting of the forward oxidation of methane and the forward and backward carbon monoxide oxidation. One limitation of the model used in the 3D single vane burner was found while examining the laminar flame. This limitation was due to the concentration exponent for methane of -0.3, found in equation (5.6). When using the combustion model with a -0.3 concentration exponent the rate of methane consumption increases, without limit, as the methane concentration approaches zero.[18] This limitation did not present itself in the 3D turbulent simulation because FLUENT limits the reaction rates by using turbulent rates.[4] Therefore, if the Arrhenius calculated reaction rates reached exceptionally high values, they are limited by assuming that the overall rate of reaction is dictated by turbulence mixing. Additional information can be obtained from the Eddy-Dissipation Model defined in the FLUENT users guide.[4] The major implication of the limitation of the combustion model used for the 3D problem is that it required a modified combustion model be used for the simulation of the Sandia laminar flame. This is because there is no turbulent mixing rate to help limit the reaction rate. This modified methane/air

reaction mechanism is presented below.

$$\begin{gathered} CH_4 + 1.5O_2 \rightarrow CO + 2H_2O \\ CO + 0.5O_2 \leftrightarrows CO_2 \end{gathered} \tag{6.1}$$

$$\begin{gathered} k_1 = A_1 \exp\left(E_1/R/T\right) \left[CH_4\right]^{0.7} \left[O_2\right]^{0.8}, \\ A_1 = 5.012 \cdot 10^{11}\ s^{-1}, E_1/R = 24054K \end{gathered} \tag{6.2}$$

$$\begin{gathered} k_{2f} = A_{2f} \exp\left(E_{2f}/R/T\right) \left[CO\right] \left[O_2\right]^{0.25} \left[H_2O\right]^{0.5} \\ A_{2f} = 2.239 \cdot 10^{12} \left(m^3/Kmol\right)^{0.75} s^{-1}, E_{2f}/R = 20807K. \end{gathered} \tag{6.3}$$

$$\begin{gathered} k_{2b} = A_{2b} \exp\left(E_{2b}/R/T\right) \left[CO_2\right]^1 \\ A_{2b} = 5.0 \cdot 10^{8} \left(m^3/Kmol\right)^{0.75} s^{-1}, E_{2b}/R = 20807K. \end{gathered} \tag{6.4}$$

This combustion model is the default methane/air two-step reaction mechanism found in FLUENT.

C. Numerical Model

Much of the numerical model has already been presented in the numerical model chapter, which outlines how the flow solver works. However, there are some additional numerical details specific to this problem that will be introduced here. This section will discuss the additional numeric specifics used to simulate Flame A of the Sandia Diffusion Flame facility.

First, the boundary conditions will be discussed. Then the grid generation will be presented. Along the way, meshes which were not as successful will be discussed and an explanation for their deficiencies included. This will be included as it seems important to document some of the lessons learned from this research. For purposes of stability, a temperature limit was placed on the numerical simulation. The reasons

this limit was imposed will be discussed.

1. Boundary Conditions

The experiment described the entire flow domain as isobaric. That is probably not exactly, or strictly true, because the flow must have some driving force. Because the test was described as an isobaric condition, it was the objective of the simulation to ensure that the simulation has only minimal, if any, variations in pressure. Experimental flow properties were given in terms of velocities, for the co-flow and mass flow rate for the premixed fuel injection. Therefore, velocity inlet boundary conditions were used in the simulations. The inlet boundaries were necessary to model the co-flow as well as premixture of air and fuel. The velocity inlet boundary condition specifies the magnitude and direction of the velocity. The static temperature of the flow into the domain is also specified. Since a combustive flow is being simulated, species mass fractions must be specified at the inlet boundary as well. The static pressure then slightly adjusts to accommodate the prescribed velocity distribution. In this simulation, the final solution shows that the pressure is essentially constant at the inlet of the co-flow, the inlet of the tube, and everywhere else in the domain.

Table XIII shows the inlet boundary conditions specified for the co-flow and premixed fuel of the laminar methane flame.

Similar to the 3D injector problem, the outlet boundary conditions were specified using the static pressure. Assigning the value of static pressure to standard atmospheric conditions was the method used to model all of the stagnant boundaries for the problem. This works well because the pressure as well as the concentration of gases is known for the ambient air. As mentioned before, additional quantities are needed to allow the user to specify certain back-flow quantities, in the event that back-flow occurs in the domain. In this simulation it became important to specify

Table XIII. Sandia velocity inlet boundary conditions specification

Parameter	Premixed Fuel	Co-flow
Velocity magnitude $[m/s]$	2.90	0.4
Static Temperature $[K]$	300	300
Flow Direction	Axial†	Axial†
Mass Fraction		
CH_4	0.1527	0.000
O_2	0.1944	0.2295
CO_2	0.0004	0.0005
CO	0.000	0.000
H_2O	0.0066	0.0078
N_2	0.6459	0.7491

† denotes the x-axis direction or along the axis of the tube

these parameters because along the sides of the domain flow would sometimes travel back inside the domain. Luckily, the flow conditions were already known in places where back-flow occurred. This is because back-flow always occurred at the boundaries with ambient air. The actual quantities specified can be found in table XIV. They are the same values specified for air at standard atmospheric conditions.

The next boundary condition used was an axisymmetric boundary along the centerline of the tube and extending far downstream. The last set of boundary conditions were no-slip, adiabatic wall boundaries applied to the tube itself.

Table XIV. Sandia pressure outlet boundary conditions specification

Flow Parameter	Outer Domain
Static Pressure Pa	101325
Back-flow Total Temperature K	300
Back-flow Flow Direction	normal to boundary
Mass Fraction	
CH_4	0.000
O_2	0.2295
CO_2	0.0005
CO	0.000
H_2O	0.0078
N_2	0.7491

† denotes the x-axis direction or along the axis of the tube

2. Description of Computational Grid

Grid generation for this particular simulation required some special consideration. The key features and considerations which are necessary for the computational domain will be discussed.

Some amount of planning was needed in order to create the computational domain for this problem. There were a few iterations performed on the generation of the grid before a suitable grid was obtained. Therefore the grid was being developed at the same time the solutions were being performed and some of the solutions that were obtained dictated the next generation of the mesh.

As mentioned before, the experimental tests showed that the problem was axisymmetric. To take advantage of this, a cut was made along the center line of

the tube and extending far downstream. This cut was then set as an axisymmetric boundary, so that the plane being modeled could be wrapped 360 degrees around the axisymmetric boundary to yield a fully three-dimensional solution.

The first evolution in the creation of the grid was to define an inlet boundary that intersected the grid at the point where the tube injecting the fuel ended. Because of the axisymmetric boundary, only half of the total diameter of the hole was modeled. Therefore, the co-flow region was modeled as 6 inches past the line segments that modeled the hole and end tube wall. This grid had the appearance of a simple rectangular grid, with a segment of one of the sides set as a velocity inlet for the fuel, and another segment set as an inlet boundary for the co-flow air. In addition to these boundaries, an axisymmetric boundary was used along with a wall boundary condition to model the tube. The remainder of the boundaries were defined as pressure outlet boundaries. These boundaries are shown in figure 29. The actual parameters specified at the boundary were described in the preceding section.

There was one major problem with the first generation grid. Figure 30 shows a temperature contour plot where a grid boundary is located at the exit of the tube. Also it is important to note that figure 30 shows a contour plot where the axisymmetric boundary has been mirrored and the entire figure rotated 90° clockwise so that more of the details along the axial direction are visible. Placing the boundary at the exit of the tube, which is where the premixed fuel is being injected, did not allow the flame to set up correctly. This is mainly due to the fact that the boundary was being described as if it had a constant velocity all along the fuel inlet domain. In actuality, a boundary layer develops along the length of the tube, both inside and outside, and this boundary layer affects the shape of the flame that results. Thus it became necessary to either provide better boundary conditions, or move the boundaries further upstream so that the boundary layer can develop along the tube. The latter was chosen for

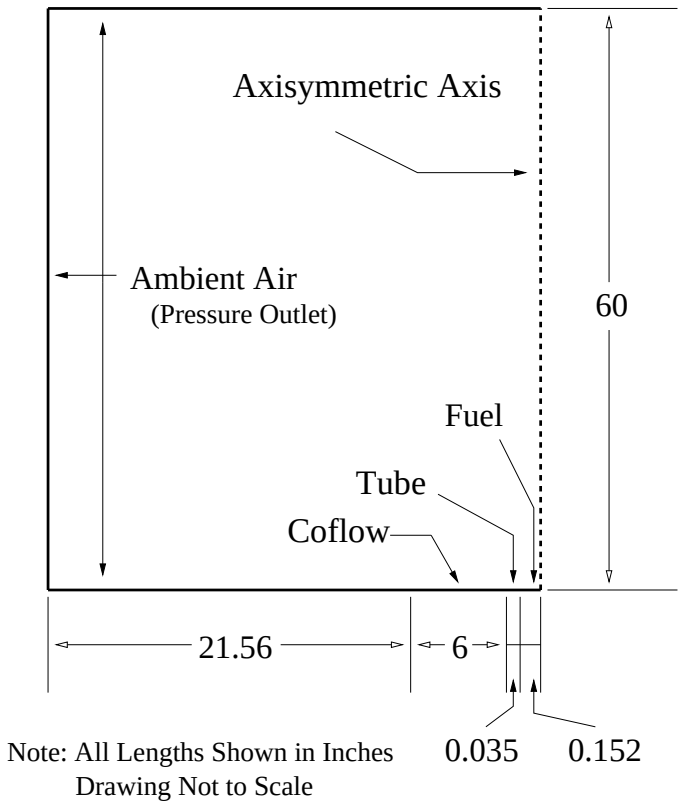


Fig. 29. Idealized pictorial description of initial laminar flame grid.

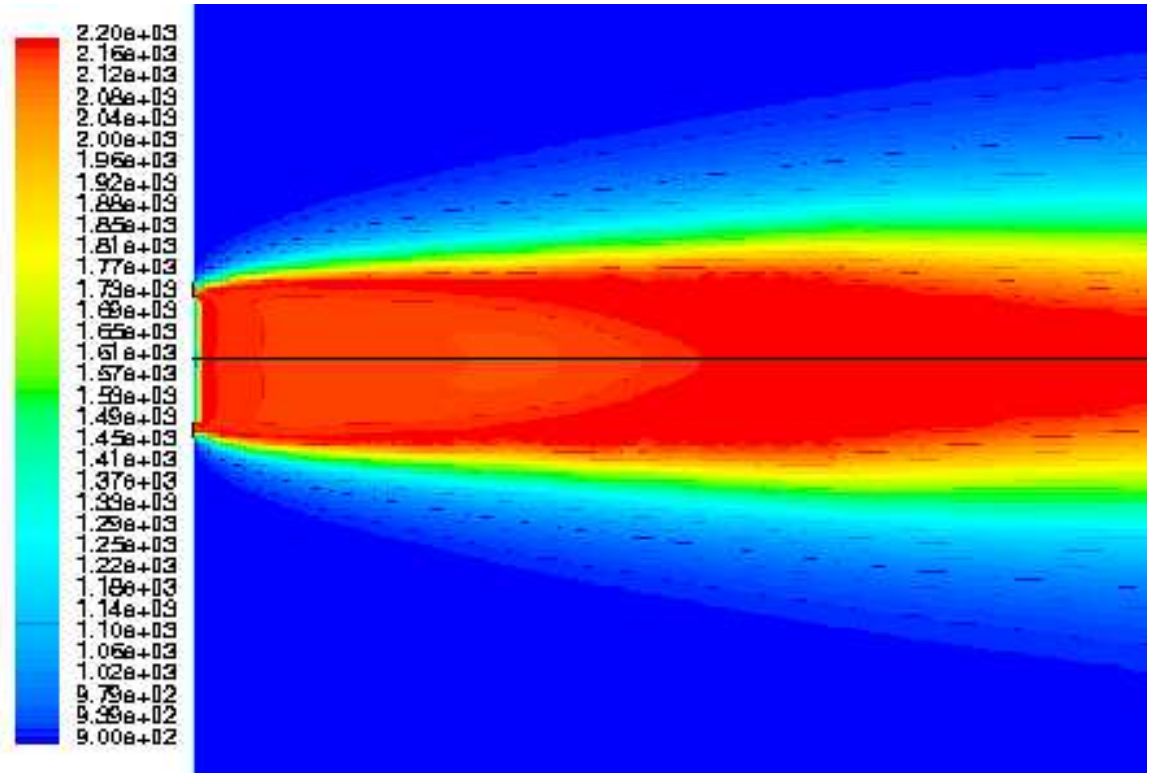


Fig. 30. Temperature contour plot with boundary located at the exit of the tube.

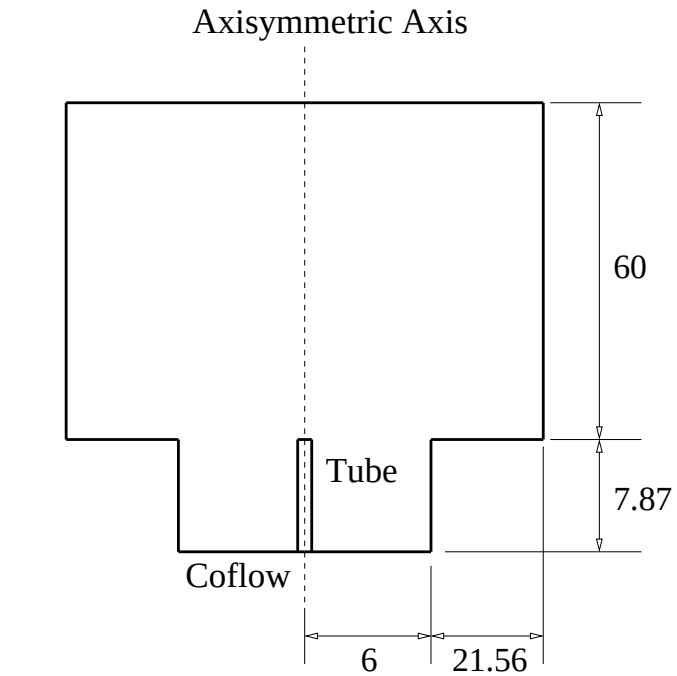


Fig. 31. Idealized illustration of final numerical domain of laminar flame simulation.

this research. Moving the grid boundaries upstream resulted in a new design of the numerical domain which allows the boundary layer region to develop and helps give a more accurate flame shape.

Another problem with the grid used to create figure 30 was a large temperature gradient close to the boundary, causing the solution to vary greatly right after the premixed gases enter the domain. This boundary significantly affected the numerical simulation obtained, so another design iteration for the mesh had to be pursued.

The new idea behind the design of the grid can be seen in figure 31. The inlet boundaries of the first generation grid were moved back 7.87 *in* to meet the exit of the wind tunnel that provides the 0.4 *m/s* co-flow. This was done for the boundaries that described the inlet of the tube, the inlet of the co-flow and the wall of the tube, creating the rectangular region at the bottom of figure 31. For clarity, the figure shows

a second half of the numerical domain which acts as a two dimensional cut of the actual three-dimensional space. However, the numerical simulation only uses half of the domain shown in figure 31 and then mirrors the solution across the axisymmetric boundary. One approximation that results in this formulation is that the square twelve inch vertical wind tunnel which is the source of the co-flow is actually modeled as a 12 *in* diameter circular wind tunnel. This simplification should have minimal effects on the numerical solution.

Another important consideration necessary in the creation of the grid was how to initiate combustion. Because the temperature used for the inlet boundary conditions of the co-flow and premixed gas were both at $300K$, there is no spark to get the chemical reaction started. In addition, FLUENT does not allow access to specific cells within the domain, so a small region had to be created in the grid using GAMBIT. In actuality many such small regions were constructed along locations where the premixed fuel exits the tube. This allowed the user to artificially begin the combustion process by patching all of the cells of a specific region with a temperature value high enough to begin combustion.

A grid convergence study was performed which compared the results of three different levels of refinement to the experimental data. The final grid chosen contained 167,523 cells. In addition to regions of grid cells constructed to control the numerical spark, other regions were constructed to control the spacing of the cells throughout the domain. Grid nodes were clustered along the length of the flame to provide a finer mesh in this area. The final mesh consisted of both unstructured and structured elements. A structured boundary layer mesh was created along the tube to capture the boundary layer effects. This was important because the resolution of the boundary layer does affect the shape of the resulting flame. Wall y^+ never exceeded 0.5 along the length of the tube.

3. Temperature Limiter

During the initial laminar simulations, it was found that the solution would diverge. Initially, this was due to many factors. In the beginning, the combustion model used was the same as the model B used in the 3D vane burner. However, this model had inherent numerical difficulties discussed earlier in this chapter. Also, it was difficult to initiate combustion. In the single vane burner, the temperature of the gases entering the burner was already of high temperature, about $1500K$. Thus, when the numerical switch that allows for combustion simulation was turned on, combustion immediately took place. In this simulation, the temperature of the co-flow and premixture is set at $300K$. Therefore, when combustion is turned on, the gases do not combust automatically because the temperature levels are not high enough to initiate it. In order to overcome this, a numerical spark was necessary. To do this, the numerical domain shown in figure 31 was broken into small regions. Then one of these small regions located at the exit of the tube was assigned temperature values of $1500K$. This provided the numerical spark necessary to initiate combustion for the laminar flame.

Unfortunately, even when using the combustion model presented in this chapter, that proves to be more inherently stable than the one used for the 3D single vane burner, the solution would still diverge. In order to keep this solution bounded, especially in the early stages of the combustion solution process, it was necessary to limit the maximum temperature value which was stored in a given cell in the domain. As the numerical solution neared convergence, the temperature limiter could be relaxed. Unfortunately, we found that the value chosen for the temperature limiter affects the solution. This is especially bad because the use of this temperature limiter is not supported by physical reasoning. However, its use is necessary to obtain a

converged solution, at least when this combustion model and numerical model are used together.

To check how the solution changed with temperature limiter, five cases were run. The five different temperature limits were $2025K$, $2300K$, $2600K$, $2900K$ and $5000K$. Numerical simulations were run with each and the results will be discussed later.

D. Entropy Inequality Expression with Numerical Implementation

The entropy inequality, or second law of thermodynamics, written for multi-component reacting mixture can be represented by the following equation[8] :

$$-tr[(\overline{\overline{T}}+P\overline{\overline{I}})\cdot\nabla\vec{v}]+cRT\sum_{B=1}^{N-1}\vec{J}_B\cdot\left(\frac{\vec{d}_B}{\rho_B}-\frac{\vec{d}_N}{\rho_N}\right)+\sum_{B=1}^{N-1}(\mu_B-\mu_N)r_B+\frac{1}{T}\vec{\epsilon}\cdot\nabla T\leq 0. \quad (6.5)$$

Each term of this expression will be discussed in more detail in what follows. It is important to say that N is equal to the number of chemical species in the multi-component mixture. For much of this work, $N = 6$, and the species are $H_2O, CO_2, N_2, CH_4, CO$ and O_2. The left-hand side of inequality (6.5) is calculated in FLUENT using a user defined function (UDF).

1. First Term

The first term represents the amount of entropy increase due to shear stresses in the domain. tr represents the trace of a matrix. The stress tensor $\overline{\overline{T}}$ is given for a Newtonian fluid as

$$\overline{\overline{T}} = (-P+\lambda(\nabla\cdot\vec{v}))\overline{\overline{I}}+2\mu\overline{\overline{D}}. \quad (6.6)$$

P is defined as the thermodynamic pressure, λ is defined as the bulk viscosity and is approximated by

$$\lambda = -\frac{2}{3}\mu. \tag{6.7}$$

$\nabla \cdot \vec{v}$ is the divergence of the velocity, $\overline{\overline{I}}$ is the identity matrix and μ is the shear viscosity. $\overline{\overline{D}}$ is described as the rate of deformation tensor and is defined as

$$\overline{\overline{D}} \equiv \frac{1}{2}[\nabla\vec{v} + (\nabla\vec{v})^T]. \tag{6.8}$$

The substitution of these equations into the first term yields,

$$-tr\left[\left(- P\overline{\overline{I}} - \frac{2}{3}\mu(\nabla\cdot\vec{v})\overline{\overline{I}} + 2\mu\frac{1}{2}[\nabla\vec{v} + (\nabla\vec{v})^T] + P\overline{\overline{I}}\right)\cdot\nabla\vec{v}\right]. \tag{6.9}$$

After some simplification the first term in the entropy inequality is written as

$$-tr\left[\left(\mu[\nabla\vec{v} + (\nabla\vec{v})^T] - \frac{2}{3}\mu(\nabla\cdot\vec{v})\overline{\overline{I}}\right)\cdot\nabla\vec{v}\right]. \tag{6.10}$$

2. Second Term

The second term represents entropy created due to diffusion of species. c is the total molar density of the flow at a given point in the domain. R is the universal gas constant and T is the temperature. The total molar density is given by the expression[6]

$$c = \sum_{i=1}^{N} c_i \tag{6.11}$$

and c_i is given by[6]

$$c_i = \frac{\rho_i}{M_i}. \tag{6.12}$$

ρ_i is the density of the i^{th} species in the solution at a given location and M_i is the molecular mass of the i^{th} species.

$\vec{J_i}$ is the i^{th} species diffusive mass flux vector. This term has already been explained in chapter II but will be presented again for completeness. Its expression is given by,

$$\vec{J_i} = -\sum_{j=1}^{N-1} \rho D_{ij} \nabla Y_j - D_{T,i} \frac{\nabla T}{T} \tag{6.13}$$

where Y_j is the mass fraction of species j. Other terms which compose the diffusive mass flux vector are defined as follows:[28,4]

$$[D] = [A]^{-1}[B] \tag{6.14}$$

$$D_{ij} = [D] = [A]^{-1}[B] \tag{6.15}$$

$$A_{ii} = -\left(\frac{X_i}{\mathcal{D}_{iN}} \frac{M_{mix}}{M_N} + \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{X_j}{\mathcal{D}_{ij}} \frac{M_{mix}}{M_i} \right) \tag{6.16}$$

$$A_{ij} = X_i \left(\frac{1}{\mathcal{D}_{ij}} \frac{M_{mix}}{M_j} - \frac{1}{\mathcal{D}_{iN}} \frac{M_{mix}}{M_N} \right) \tag{6.17}$$

$$B_{ii} = -\left(X_i \frac{M_{mix}}{M_N} + (1 - X_i) \frac{M_{mix}}{M_i} \right) \tag{6.18}$$

$$B_{ij} = X_i \Big(\frac{M_{mix}}{M_j} - \frac{M_{mix}}{M_N} \Big) \tag{6.19}$$

$$M_{mix} = \sum_{i=1}^{N} M_i X_i. \tag{6.20}$$

The matrices $[A]$ and $[B]$ are $(N-1)$x$(N-1)$ in size. X_i is the species mole fraction. M_{mix} is the molecular mass of the mixture at a given location and N stands for the number of species in the mixture. When N is used as a subscript it means that it is the N^{th} species in the mixture. $\mathcal{D}_{ij}$ is taken to be the binary mass diffusion coefficient. The binary mass diffusion coefficients are calculated using the following method[29,4]:

$$\mathcal{D}_{ij} = 0.0188 \frac{\left[T^3 \left(\frac{1}{M_i} + \frac{1}{M_j}\right)\right]^{1/2}}{P_{abs} \sigma_{ij}^2 \Omega_D}, \tag{6.21}$$

where P_{abs} is the absolute pressure, and Ω_D is the diffusion collision integral, which is a measure of the interaction of the molecules in the system. Ω_D is a function of the quantity T^*.

$$\Omega_D = [A(T^*)^{-B}] + C[e^{-DT^*}] + E[e^{-FT^*}] \tag{6.22}$$

In the above formulation for the diffusion collision integral, $A = 1.16145, B = 0.14874, C = 0.52487, D = 0.77320, E = 2.16178, F = 2.43787$, and T_D^* is given by the following expression

$$T^* = \frac{T}{(\epsilon/k_B)_{ij}}. \tag{6.23}$$

k_B is the Boltzmann constant, which is defined as the gas constant, R, divided by Avogadro's number.[5] $(\epsilon/k_B)_{ij}$ is a geometric average of the parameters for the individual species.[29]

$$(\epsilon/k_B)_{ij} = \sqrt{(\epsilon/k_B)_i(\epsilon/k_B)_j} \tag{6.24}$$

For a binary mixture, σ_{ij} is calculated as the arithmetic average of the individual σ values.[29]

$$\sigma_{ij} = \frac{1}{2}(\sigma_i + \sigma_j) \tag{6.25}$$

σ_i and $(\epsilon/k_B)_i$ are defined as the Lennard-Jones parameters, and their values are defined in tabulated form.[5]

This defines all of the necessary values needed to calculate the diffusive mass flux vector except for the thermal diffusion coefficient, $D_{T,i}$. The method used to calculate, $D_{T,i}$, is an empirically based, composition dependent expression given by[4]

$$D_{T,i} = -2.59 \times 10^{-7} T^{0.659} \left[\frac{M_i^{0.511} X_i}{\sum_{i=1}^{N} M_i^{0.511} X_i} - Y_i \right] \cdot \left[\frac{\sum_{i=1}^{N} M_i^{0.511} X_i}{\sum_{i=1}^{N} M_i^{0.489} X_i} \right] \tag{6.26}$$

With this information the diffusive mass flux can be calculated, but there is still an important parameter in the second term of the entropy inequality, equation (6.5), which still must be explained. For ideal gases $\vec{d}_B$ has the following expression,[8] where the subscript B is changed to i to make it more general for any specific species within the domain.

$$\vec{d}_i = \nabla X_i + \frac{X_i M_i}{RT} \left[\left(\overline{\nabla}_i - \frac{1}{\rho} \right) \nabla P - \vec{f}_i + \sum_{j=1}^{N} Y_j \vec{f}_j \right]. \tag{6.27}$$

The experimental tests showed that the entire problem which is being investigated in this simulation is isobaric, and numerical simulations show practically the same. Therefore, if the entire domain is isobaric, then the pressure gradient anywhere inside the domain is small and therefore is assumed negligible in equation (6.27). External

forces, $\vec{f_i}$, are neglected because it is determined that no electrical forces have effects on the problem and that the force of gravity is negligibly small and affects all species the same.

In actuality, however valid or invalid these assumptions are, cannot entirely be determined. This is because FLUENT makes this assumption in its core when it calculates $\vec{d_i}$, and this cannot be overridden. So it is not possible to run a case where pressure gradient and body forces are kept in this term and compare it to a simulation without them. When making these assumptions to calculate the second law, the resulting calculation will be at least as accurate as using any other parameters from the combustive solution because values pulled from FLUENT use the same assumptions. For example, in order to calculate the second law as it is described here, the temperature at each cell of the domain is needed. The temperature is a quantity that is calculated by FLUENT, but its value is only as accurate as the assumptions in the theory that FLUENT uses. FLUENT does not account for pressure gradient or body forces in its formulation of $\vec{d_i}$ so the value of the temperature that FLUENT computes is influenced by this assumption. Simply put, the expression (6.27) reduces to

$$\vec{d_i} = \nabla X_i. \tag{6.28}$$

Once these approximations are made, everything in the second term of the entropy inequality can be calculated. However, some additional complications come from FLUENT. FLUENT allows the user to gain access to the gradient of the mass fraction, not the gradient of the mole fraction. So a relation must be used to find ∇X_A from ∇Y_A. This will be discussed in what follows.

The relationship between the mass fraction and the mole fraction can be written

as shown below.

$$Y_i = \frac{M_i}{\sum_{j=1}^{N} M_j X_j} X_i \tag{6.29}$$

Taking the gradient of both sides gives,

$$\nabla Y_i = \nabla \left(\frac{M_i}{\sum_{j=1}^{N} M_j X_j} X_j \right). \tag{6.30}$$

Expanding the expression yields

$$\nabla Y_i = M_i \left(\nabla X_i \frac{1}{\sum_{j=1}^{N} M_j X_j} - \frac{X_i \sum_{i=j}^{N} M_j \nabla X_j}{\left(\sum_{j=1}^{N} M_j X_j \right)^2} \right). \tag{6.31}$$

Further simplification results in the following expression

$$\frac{\nabla Y_i M_{mix}}{M_i} = \nabla X_i - \frac{X_i \sum_{j=1}^{N} M_j \nabla X_j}{M_{mix}}. \tag{6.32}$$

The unknowns are the gradients of the mole fraction, that generates a system of $N-1$ equations. There are $N-1$ equations because of the property of the mole fraction and mass fraction, $\sum_i^N Y_i = 1$ and $\sum_i^N X_i = 1$. In this way the gradient of the mole fraction of the last species is written in terms of the gradient of the mole fraction of the first $N-1$ species, or

$$\nabla X_N = -\sum_{i=1}^{N-1} \nabla X_i. \tag{6.33}$$

After solving the system of equations all terms are known which are needed to calculate the second term in the entropy inequality or equation (6.5).

3. Third Term

The third term from equation (6.5) represents the entropy created due to chemical reaction, which is important in combustive flows. As the third term appears in (6.5) it is difficult to evaluate. The third term in the entropy inequality can be expressed as[8]

$$RT\sum_{r=1}^{K}\sum_{i=1}^{N}\ln\left(\frac{1}{K_r}\left[\gamma_i X_i\right]^{\nu_{i,r}}\right)\frac{r_{N,r}}{M_N\nu_{N,r}}. \tag{6.34}$$

An explanation of the terms used in the expression starts with R, which is the universal gas constant and T, which is the thermodynamic temperature. The remaining terms require a more thorough explanation. K_r is the reaction equilibrium constant for reaction r. This quantity can be calculated for each reaction which is being used to model the combustion process.

It is very important to compute the equilibrium constant for a specified reaction at any point in the domain. Because the temperature change is so large, an expression for the equilibrium constant as a function of temperature is required. An equilibrium constant is needed for each reaction, however the introduction of the expression for the equilibrium constant is given here in a general format. The equilibrium constant is related to the Gibbs energy change of reaction by[30]

$$\frac{\Delta G^\circ}{RT} = -\ln K. \tag{6.35}$$

Also, the standard heat of reaction can be related to Gibbs energy change when using

$$\Delta H^\circ = -RT^2\frac{d(\Delta G^\circ/RT)}{dT} \tag{6.36}$$

Therefore we obtain

$$\frac{d(\ln K)}{dT} = \frac{\Delta H^\circ}{RT^2}. \tag{6.37}$$

This equation gives the effect of temperature on the equilibrium constant. If the standard heat of reaction is known as a function of T, then (6.37) can be integrated to yield,

$$\ln K = \int \frac{\Delta H^\circ}{RT^2} dT + I \tag{6.38}$$

where, I, is a constant of integration. The general expression of ΔH is

$$\Delta H^\circ = J + \int \Delta C_p^\circ dT, \tag{6.39}$$

where J is another integration constant. When each C_p is approximated as an ideal gas specific heat it can be written as

$$\frac{C_p^{ig}}{R} = A + BT + CT^2 + DT^{-2} \tag{6.40}$$

The expression resulting from equation (6.39) and (6.40) is

$$\frac{\Delta H^\circ}{R} = \frac{J}{R} + (\Delta A)T + \frac{\Delta B}{2}T^2 + \frac{\Delta C}{3}T^3 - \frac{\Delta D}{T} \tag{6.41}$$

Substitution of this expression into (6.38) and integration gives

$$\ln K = -\frac{J}{RT} + (\Delta A)\ln T + \frac{\Delta B}{2}T + \frac{\Delta C}{6}T^2 + \frac{\Delta D}{2T^2} + I. \tag{6.42}$$

Further details of this derivation are found in [Smith and Van Ness].[30] A, B, C, and D are constants used to compute the heat capacities of gases in an ideal gas. These constants are obtained from tabulated sources.[30]

$$\Delta A = A_{(\text{Products})} - A_{(\text{Reactants})} \tag{6.43}$$

The Δ operator is explained as the A values of the reactants subtracted from the A values of the products for the specific reaction being investigated. Evaluation of the constants J and I requires standard values of ΔH_{298} and ΔG_{298}. Once these constants are found for the specific reaction, all information is known in order to calculate the equilibrium constant as a function of temperature.

The next term which requires attention is γ_B, the activity coefficient defined as

$$\gamma_i \equiv \frac{a_i}{x_i} \tag{6.44}$$

For an ideal gas $\gamma = 1$ so it becomes necessary to check the assumption of ideal gas for this problem using the compressibility factor. The main components in the system are air and methane. Both enter into the domain at room conditions but during combustion room temperature no longer becomes a valid assumption. However, the problem is assumed to be practically isobaric, as was verified in the experiment and shown in the simulation. Thus in calculating the reduced pressure for methane we get

$$P_R = \frac{P}{P_{cr}} = \frac{1.013E5 \ \ Pa}{4.64E6 \ \ Pa} = 0.0216, \tag{6.45}$$

and calculating the reduce pressure for air we obtain

$$P_R = \frac{P}{P_{cr}} = \frac{1.01e5 \ \ Pa}{3.771e6 \ \ Pa} = 0.02687. \tag{6.46}$$

From the generalized compressibility chart the compressibility factor is very close to unity.[5] So it is believed that the flow in this problem behaves very much like and ideal gas, particularly due to the very low reduced pressure in this problem. This

means that γ is set equal to 1.

A one-step chemical reaction of arbitrary complexity can be represented by the following stoichiometric equation:

$$\sum_{i=1}^{N} \nu_i^{'} S_i \rightarrow \sum_{i=1}^{N} \nu_i^{''} S_i. \tag{6.47}$$

S is an arbitrary specification of the chemical species, $\nu_i^{'}$ and $\nu_i^{''}$ are the stoichiometric coefficients for the reactants and products, respectively, and N is the total number of chemical species in the one-step reaction. An example which shows this notation is written as:

$$CH_4 + 1.5O_2 \rightarrow CO + 2H_2O, \tag{6.48}$$

where

$$\begin{array}{llll} S_1 = CH_4, & S_2 = O_2, & S_3 = CO, & S_4 = H_2O, \\ \nu_1^{'} = 1, & \nu_2^{'} = 1.5, & \nu_3^{'} = 0, & \nu_4^{'}=0, \\ \nu_1^{''} = 0, & \nu_2^{''} = 0, & \nu_3^{''} = 1, & \nu_4^{''}=2. \end{array}$$

The $\nu_{i,r}$ that appears twice in equation (6.34) is defined as the generalized stoichiometric coefficient. It is defined as

$$\nu_{i,r} = \nu_{i,r}^{''} - \nu_{i,r}^{'} \tag{6.49}$$

Therefore, the generalized stoichiometric coefficient is simply the difference between the stoichiometric coefficient of the product and the reactant. A generalized stoichiometric coefficient is defined for each species i in each reaction r.

The mass rate of production of species i by chemical reaction r is written as $r_{i,r}$. It can be calculated by

$$r_{i,r} = M_i \hat{R}_{i,r}. \tag{6.50}$$

M_i is the molecular mass of the species i and $\hat{R}_{i,r}$ is the Arrhenius molar rate of creation or destruction of species i in reaction r. Thus it is important to re-emphasize that the subscript i denotes which species is being effected, and the subscript r describes in which reaction that species is being created or destroyed.

The molar rate of creation/destruction of species i in reaction r is given by[4,6]

$$\hat{R}_{i,r} = \left(\nu''_{i,r} - \nu'_{i,r}\right)\left(k_{f,r}\prod_{j=1}^{N_r}[C_{j,r}]^{\eta'_{j,r}} - k_{b,r}\prod_{j=1}^{N_r}[C_{j,r}]^{\eta''_{j,r}}\right). \tag{6.51}$$

This expression introduces many new terms and they are defined as:

N_r	=	number of chemical species in reaction r
$C_{j,r}$	=	molar concentration of each reactant and product in species j in reaction r. Typical units are $\left[\frac{Kmol}{m^3}\right]$
$\eta'_{j,r}$	=	forward rate exponent for each reactant and product species j in reaction r
$\eta''_{j,r}$	=	backward rate exponent for each reactant and product species j in reaction r
$k_{f,r}$	=	forward rate constant for reaction r
$k_{b,r}$	=	backward rate constant for reaction r
$\nu'_{i,r}$	=	stoichiometric coefficient for reactant i in reaction r
$\nu''_{i,r}$	=	stoichiometric coefficient for product i in reaction r

It is important to note that this representation of $\hat{R}_{i,r}$ does not include the net effect of third bodies on the reaction rate but they can be added when third body reactions must be modeled.

The forward rate constant for reaction r, $k_{f,r}$, is computed using the Arrhenius expression

$$k_{f,r} = A_r T^{\beta_r} e^{-E_r/RT} \tag{6.52}$$

where

A_r = pre-exponential factor

β_r = temperature exponent

E_r = activation energy for the reaction (J/Kmol)

R = universal gas constant (J/Kmol-K)

Pre-exponential factors, activation energies, stoichiometric coefficients, forward rate constants, and rate exponents are values which depend on the reaction mechanism used, but they are inputs into the simulation and can be considered "known" or "calculable" values when used to calculate the second law inequality. It is important to note that even though backward reactions were included, they were effectively included by adding an additional forward reaction which appeared identical to the backward reaction. This was done as a matter of simplifying the procedure and because more complex assumptions can be made within FLUENT.

4. Fourth Term

After calculating the first three terms, this term is relatively easy to calculate because it is composed of quantities which have already been calculated or are easily obtained from FLUENT. The temperature and gradient of temperature are quantities which can be obtained from the simulation. The term which needs explaining is $\vec{\epsilon}$. Using kinetic theory of dilute gases, $\vec{\epsilon}$ can be represented as[8]

$$\vec{\epsilon} = -k\nabla T - cRT\sum_{i=1}^{N} D_{T,i}\frac{\vec{d_i}}{\rho_i}. \tag{6.53}$$

k is the thermal conductivity, c is the total molar density, R is the universal gas constant and $D_{T,i}$ is the thermal diffusion coefficient. $\vec{d_i}$ is defined in the same manner as was used in calculation of the second term.

E. Results

The results section will be broken into two main portions. The first section of the results will discuss how well the numerical simulation compares with the experimental data, after all, that is the ultimate goal of the combustion simulation. However, some of the focus of the results will be devoted to the ability of the simulation to satisfy the second law of thermodynamics. Whether the second law is satisfied or not, the entropy inequality is a beneficial tool to examine the validity of the numerical solution of a combustion simulation using a simple reaction model.

1. Comparison with Experimental Results

Whenever a numerical simulation is performed, it is ideal to test how well the numerical simulation is performing. Ideally, the numerical simulation should give results just like the physical problem. In this case, the numerical simulation is compared against experimental data,[27] where experimental data are assumed to be as close to the actual solution as possible.

Previously, it has been mentioned that the physical experiment measured all species and temperature data at three axial locations downstream of the fuel injection. The three axial locations are located at 25, 50, and 100 *mm* downstream. The physical experiment measured species concentrations that the combustion model was

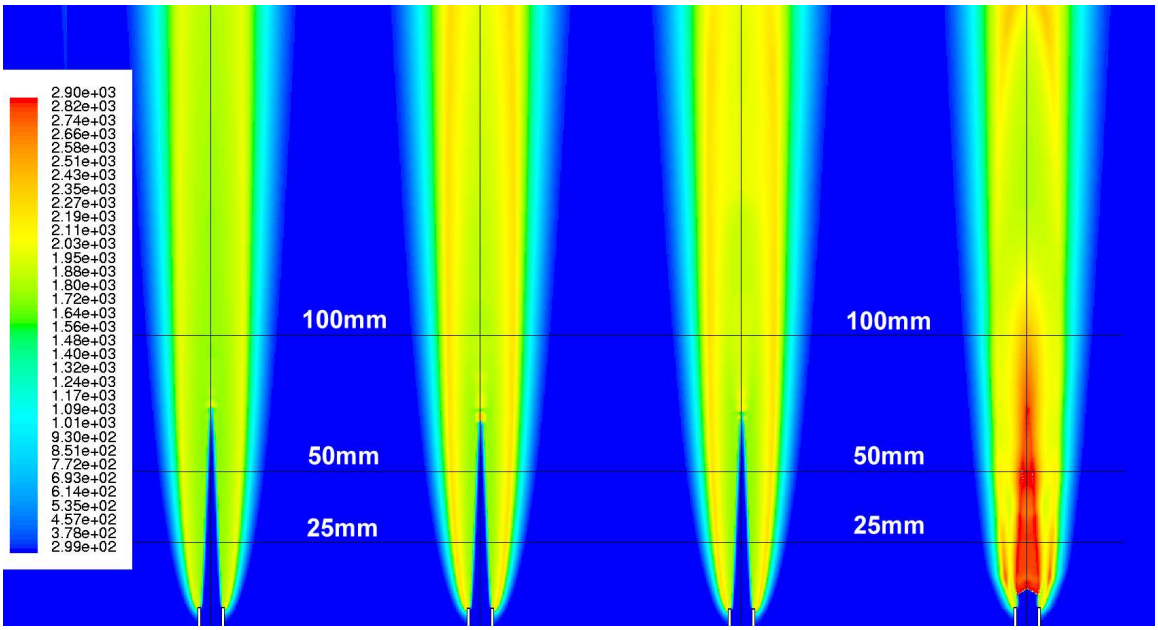


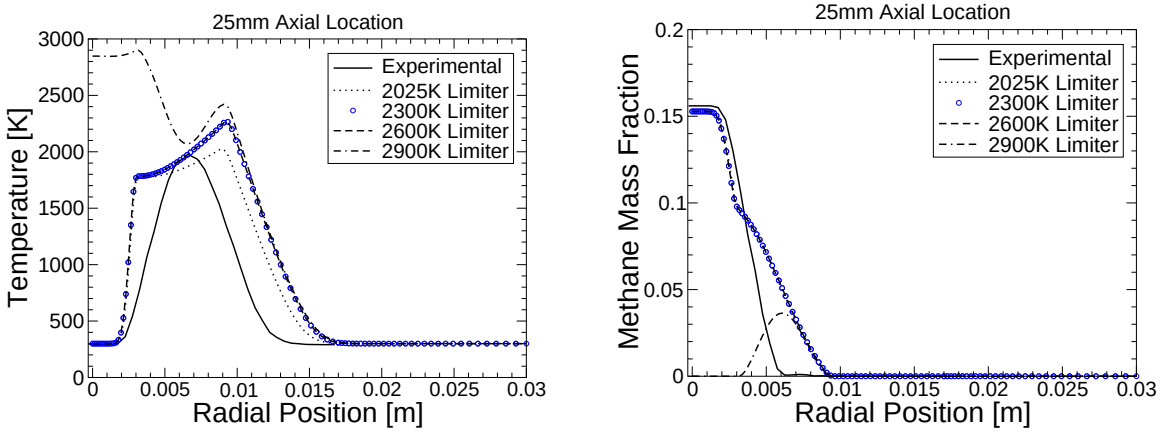
Fig. 32. Temperature contour plot for each temperature limiter. From left to right, solutions are shown for temperature limits of 2025, 2300, 2600, and 2900K.

not equipped to model. These species are NO, H_2 and OH. The following set of figures shows how the numerical simulation compares with the experimental data for varying levels of temperature limiters.

Figure 32 shows temperature contour plots of the flame region for each temperature limiter used. Therefore, four different solutions are pulled together to show the difference in the solution when using different temperature limiters. From left to right the temperature limiters are 2025, 2300, 2600, and 2900K. The temperature contour plot at the far right, the 2900K case, is not a converged solution as this contour plot was taken from a simulation only 500 iterations after the temperature limit was increased to 2900K. At 1000 iterations, the solution with a temperature limit of 2900K shows hot gases inside the tube. This suggests that combustion is taking place inside the tube, which is not in agreement with the experimental data. The same set of data used to show the temperature contour plots is used to generate the xy plots of figures 33-35.

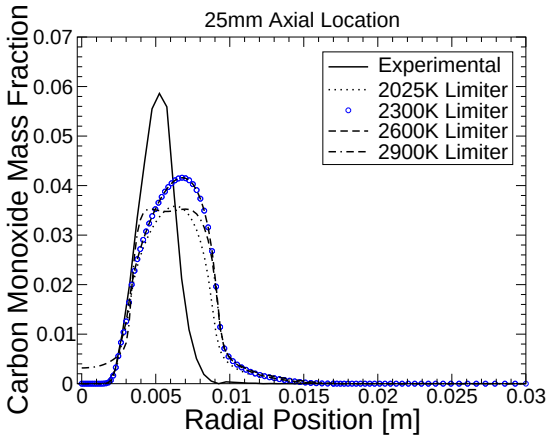
Figure 33-a shows the temperature variation along the radial direction at an axial location of 25 *mm*. Figures 33-b through 33-d show species mass fraction for certain species along the radial direction. The solid line denotes the experimental data taken at Sandia National Laboratories. The other lines are numerical solutions where a different value of the limiting temperature is used. Looking specifically at the temperature distribution along the radial direction, it is clearly seen that the numerical simulation shows a double peak that the experimental data do not show. This is due primarily to the simplified combustion model. Many of these simplified reaction schemes were developed to match flame speed, thermal distribution, or species concentrations for a specific experimental configuration.[26,24] Therefore, using such a simple combustion model in a manner other than for which it was created will probably give unsatisfactory results if all combustion parameters (flame speed, temperature distribution, species concentrations) want to be matched. Computations involving detailed chemistry are more reliable however, they come with an added computational cost.

Note from the data at 25 *mm* downstream at the axisymmetric line, or along the center line of the flame, that combustion does not occur. The first indication is the low temperature levels at the centerline. Further proof comes from looking at the species mass fractions for methane. It is seen that it is very high at the centerline and decreases as the grid location moves radially outward. Also, there is no carbon monoxide at the center line. The numerical simulation is able to capture the absence of combustion at the center line at this axial location, 25 *mm*, for all simulations except for the case with limiting temperature of $2900K$. When the simulation is performed with this temperature limiter, combustion seems to be taking place at the centerline. Overall, the simulation with a temperature limiter of $2900K$ results in a solution that is most unlike the experimental data. Looking at the other extreme,

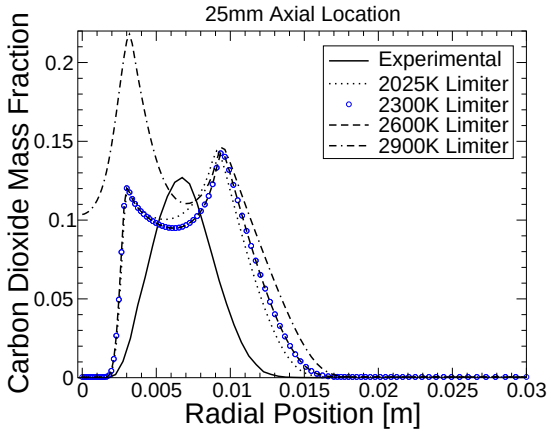


(a)

(b)



(c)



(d)

Fig. 33. Comparison with experimental data at 25 *mm*.

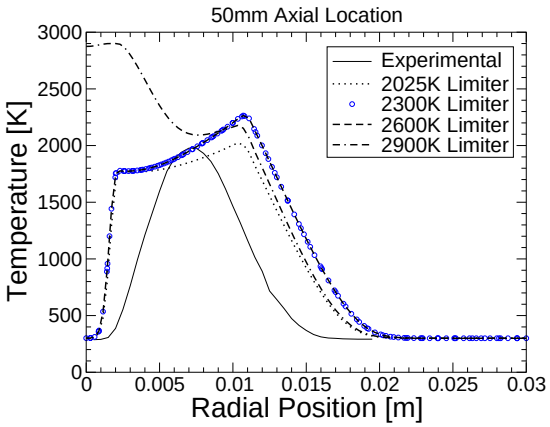
the case with a limiting temperature of $2025K$, shows arguable the best agreement with the experimental data.

When comparing with experimental data, the overall temperature magnitude is best matched with a temperature limiter of $2025K$. Also, the $2025K$ limiter captures the thickness of the flame better than the other simulations. Interestingly, the methane mass fraction shows similar variation for temperature limiters of 2025, 2300 and $2600K$. In fact, the two cases where the temperature limiter is 2300 and $2600K$, we see almost identical solutions for each of the four plots at 25 *mm*. Looking at carbon dioxide mass fraction, it is visible that the $2900K$ limiter does not capture the variation at 25 *mm*.

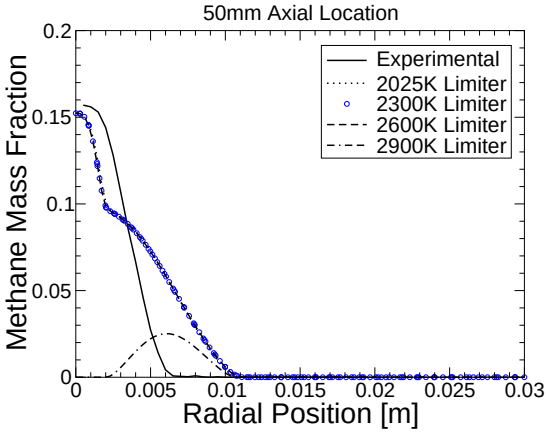
At 50 *mm* downstream, all numerical simulations except the $2900K$ limiter, capture the void of combustion right at the centerline. Another phenomenon which is seen again is the double peak region of the temperature variation. This is shown in all numerical simulations.

For the most part, the comparison between the $2900K$ temperature limiter and the other simulations shows major differences, except for the carbon monoxide variation. In terms of thickness and magnitude of the temperature variation, the simulation with $2025K$ limiter matches the experimental data best at 50 *mm*. The temperature limiters of 2300 and $2600K$ show nearly identical results for all four quantities shown.

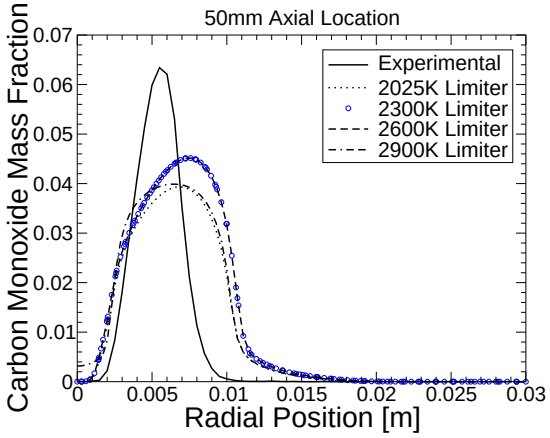
The last set of experimental data taken from the laminar flame is at 100 *mm* downstream of the fuel injection. Figure 35 shows the radial variation of temperature and methane, carbon monoxide, and carbon dioxide mass fractions. At 100 *mm* downstream, the numerical simulations no longer seem to capture the region of uncombusted gas at the axisymmetric axis. Instead, a high temperature is seen at the centerline. The most likely reason for the failure to capture even a quantitative shape of flow parameters at 100 *mm* is the use of the simplified combustion model.



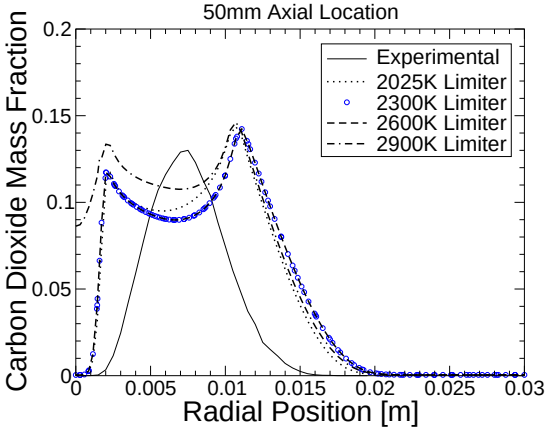
(a)



(b)



(c)



(d)

Fig. 34. Comparison with experimental data at 50 *mm*.

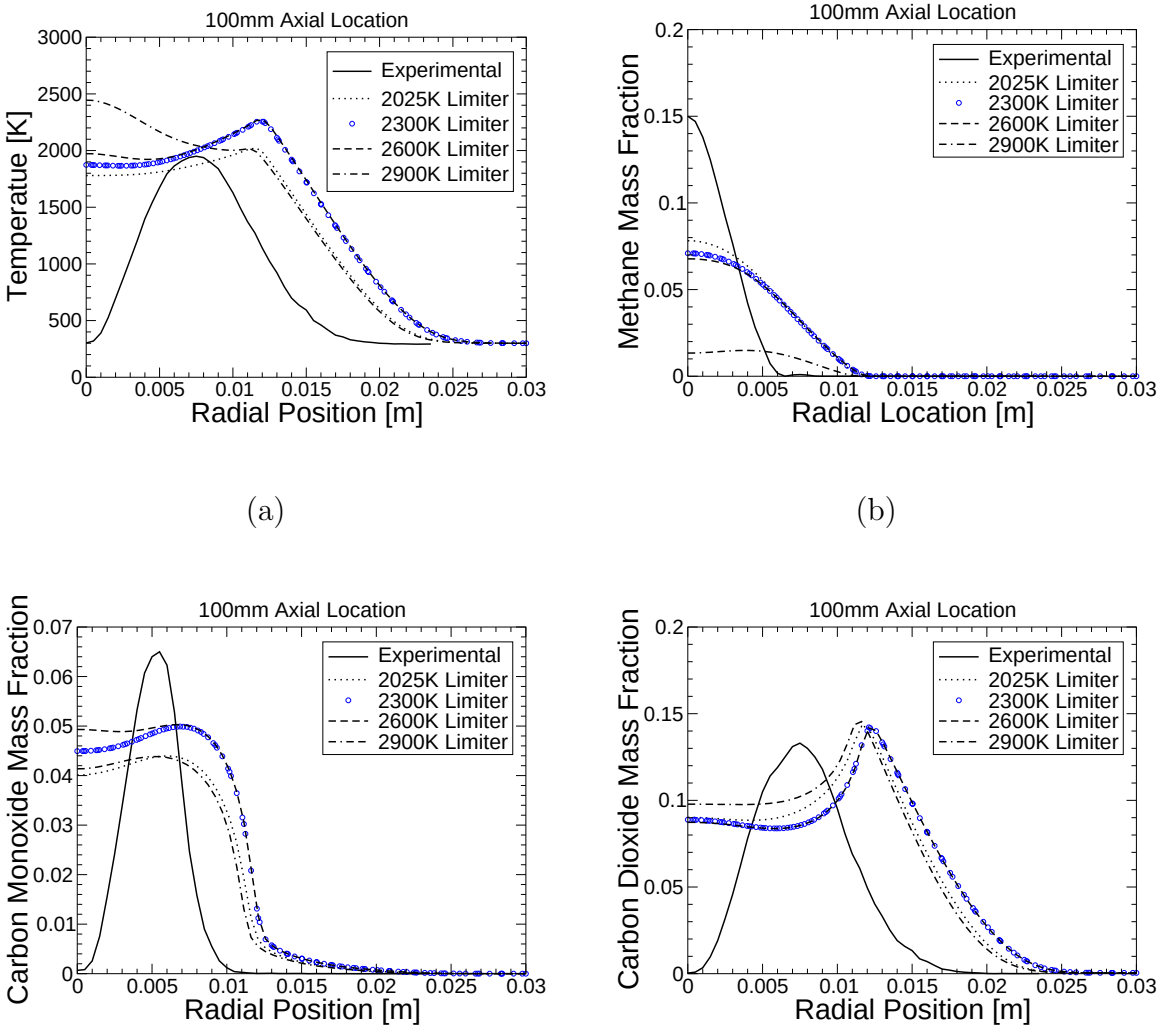


Fig. 35. Comparison with experimental data at 100 *mm*.

More robust methane/air reaction models should prove to show better comparison with experimental data. In the case with the $2900K$ limiter almost all of the methane has already been consumed at 100 mm. The remaining simulations show about half the methane mass fraction at the axisymmetric line. At 100 mm a small difference between the solution using a temperature limit of $2300K$ and $2600K$ can be seen at the axisymmetric line.

When one looks at all three axial data sets consecutively, it is seen that the radial width of the combustion region increases as the axial distance from the tube increases. This seems natural as one might expect the width of the flame to first increase as we travel axially downstream and then to decrease again as the fuel is being burned. While the numerical simulation does not capture the actual width of the flame precisely, it does manage to capture the increase in the width of the flame at the three axial locations.

An interesting result can be seen by looking at the different temperature limiters. The temperature limiter seems to limit how much energy is given off in the chemical reaction. Therefore, it is definitely affecting the solution that is obtained. This can be seen most clearly by looking at the solution using the two extreme temperature limiters. These two solutions are quite different. Also of interest is that the temperature limiter does not seem to have a drastic effect on the carbon monoxide levels in the simulations, however, it did have drastic differences in the levels of methane and temperature distribution, especially at 25 and 50 mm.

2. Entropy Inequality Results

The first step necessary in examining the entropy inequality outlined in the previous section is to look at each of the terms which are found in the entropy inequality independently. This will enable the researcher to get a physical understanding of

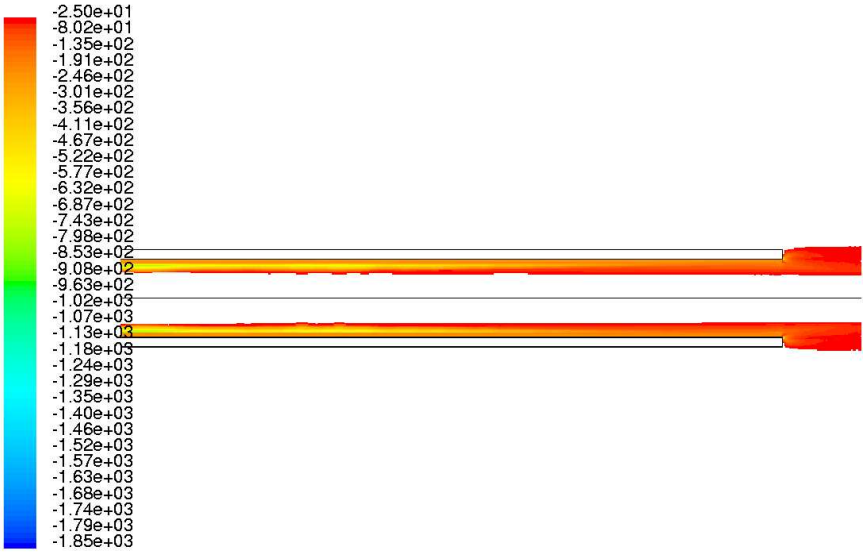


Fig. 36. Contour plot showing values of first term of entropy inequality.

each of the terms and provides a quantitative check of the calculations.

a. First Term

The first term of the entropy inequality of (6.5) is

$$-tr[(\overline{\overline{T}} + P\overline{\overline{I}}) \cdot \nabla \vec{v}]. \tag{6.54}$$

This term deals primarily with shear stresses and velocity gradients. Naturally, for this geometry, the velocity gradients and shear loads will be greatest in an around the tube. This is because the tube is the only place in this problem where a wall boundary has been declared. Figure 36 shows the points where the first term has the most negative values.

The cells with the highest negative values are places where the viscous effects are most dominant. This leads to the interpretation that the terms with the highest negative values are the places where entropy is being generated the most due to shear

stresses and velocity gradients. The overall entropy inequality given by (6.5) states that the overall expression should be a negative quantity. Thus, negative quantities are good in the sense that they tend to show that the simulation satisfies the second law of thermodynamics. For the first term, all cells have negative values for this term, however, away from the tube, the shear stress and velocity gradient are low quantities. This means that entropy generation in these regions is low. Therefore, the values of the first term away from the tube, while negative, do not have as high of a negative value, which suggests that entropy coming from the first term is not being as readily generated away from the wall boundary.

b. Second Term

The second term of the entropy inequality of (6.5) is

$$cRT \sum_{B=1}^{N-1} \vec{J}_B \cdot \left(\frac{\vec{d}_B}{\rho_B} - \frac{\vec{d}_N}{\rho_N} \right). \tag{6.55}$$

The second term shows the importance of the entropy generation due to diffusion of species. Also important in this term is the temperature in the domain, not only because it affects the thermal diffusion of the species but also because the term is multiplied by temperature. Therefore, the expectation is that this term will have the most effect where the species from the two streams, premixture and air co-flow, mix together, as well as places where the temperature is high. Looking at figure 37 it is seen that this is indeed the case. The values where the second term is most negative correspond to locations where the greatest amount of mixing as well as high temperatures occur. That location being predominately at the base of the flame region as shown by the yellow and green colors on the contour plot of figure 37. Other locations with large negative numbers extend downward where the hottest

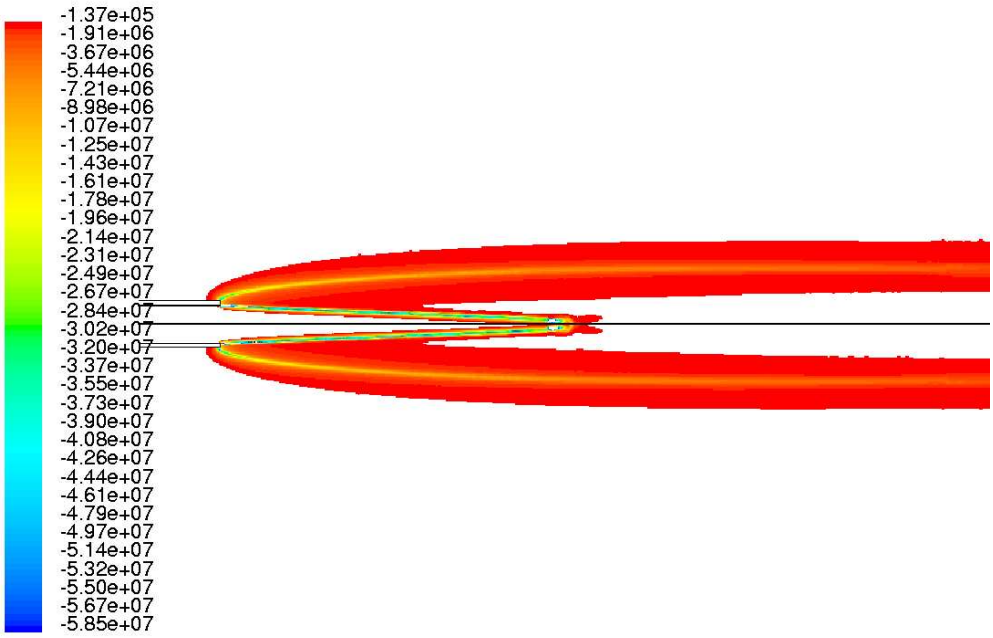


Fig. 37. Contour plot showing values of second term of entropy inequality.

regions of the flow are located. Again, the locations where the most negative cells are located are regions where entropy generation is greatest. Far away from the flame region, the second term has smaller negative numbers. This suggests that entropy due to mixing is still being generated away from the flame region, but in considerably smaller amounts.

c. Third Term

The third term in the entropy inequality has the following form:

$$RT \sum_{r=1}^{K} \sum_{i=1}^{N} \ln\left(\frac{1}{K_r}\left[\gamma_i X_i\right]^{\nu_{i,r}}\right) \frac{r_{N,r}}{M_N \nu_{N,r}}. \tag{6.56}$$

Therefore, the temperature again plays an important role. Probably the most important quantity in this term is the mass rate of production/destruction of species N by chemical reaction r, $r_{N,r}$. This term generally has extremely high rates which

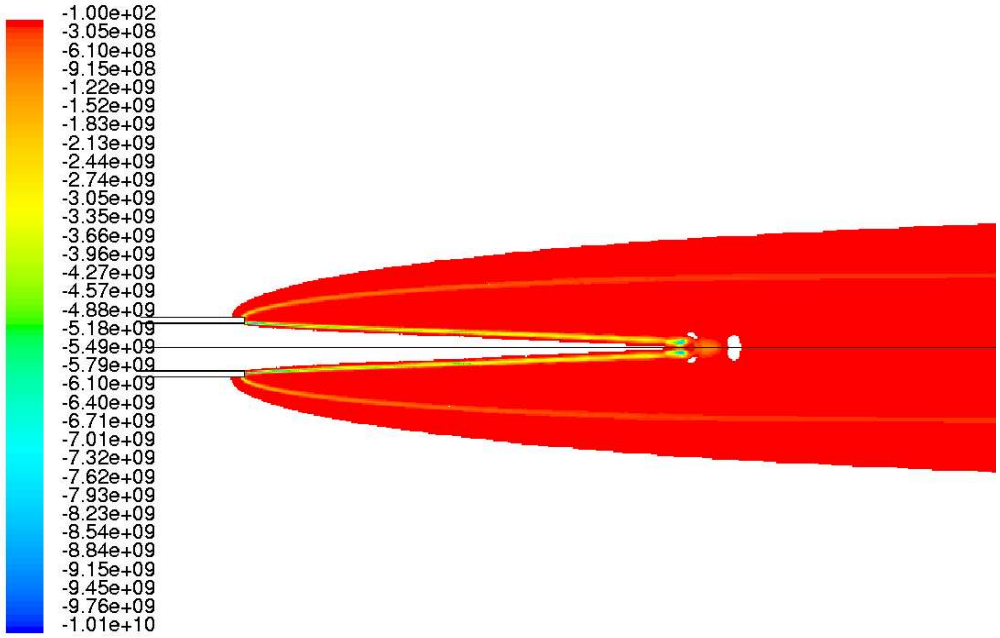


Fig. 38. Contour plot showing values of the third term of entropy inequality.

tend to dominate the values which result. Looking at figure 38 one sees that the most negative values generally occur where the temperature is high. However, one of the reasons that the temperature increases is because of the oxidation of methane which gives off heat energy and causes the temperature to rise. Therefore, regions where the temperature is greatest tend to correspond to regions where chemical reactions are taking place and thus a high rate of production/destruction. This can be seen in figure 38. The regions in this contour plot which have the highest negative numbers tend to be the regions where the reaction is occurring.

One interesting aspect of this third term is that all of the cells do not have negative values when this term is computed. Remember that negative values indicate the satisfaction of the entropy inequality. A positive value for the third term calculated at a cell does not necessarily guarantee that the second law is violated. But if the cell value has very high positive numbers, it is possible that the positive values of the

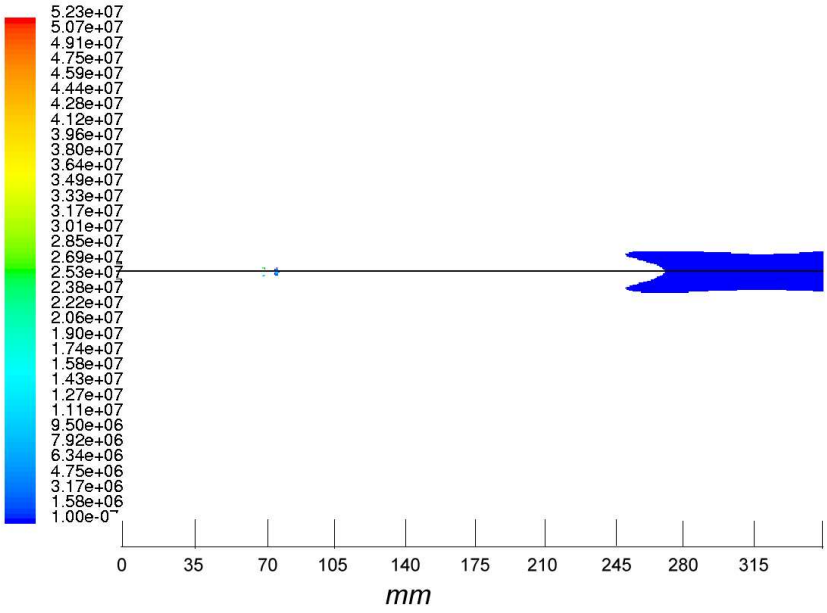


Fig. 39. Contour plot showing locations within the domain where the third term is greater than zero.

third term can overcome the negative values from the other terms and the entropy inequality can be locally violated. Figure 39 shows some of the cells in the domain that have positive values when the third term is calculated. Many of the cells with a positive calculation of the third term do not have high positive terms, as evidenced by the dominating blue contour color of figure 39. However, within this contour plot there are very small regions of relatively high positive numbers. These regions are typically very small but have positive values with magnitudes shown in the legend of figure 39. In comparison, figure 38 shows cells with high negative values for the third term and in this contour plot the regions are much larger as a wide range of colors is visible. The legends in figures 38 and 39 have different magnitudes and the cells with a high negative value are generally of higher magnitude then the cells with a high positive value.

Away from the flame region, the value of the third term is zero. This is logical

because away from the flame no chemical reactions are taking place. Therefore, the generation of entropy due to chemical reaction will be very small, or in this case, numerically zero.

d. Fourth Term

The last term of (6.5) is represented as

$$\frac{1}{T}\vec{\epsilon}\cdot\nabla T. \tag{6.57}$$

where $\vec{\epsilon}$ is represented by equation (6.54). With this representation, one sees that the thermal diffusion and temperature gradient are the dominant terms. Thus, it is expected that the highest negative numbers would occur where the temperature gradient is highest. That location is where the flame meets the co-flow region of the flow. This can be seen in figure 40, which shows the temperature contour plot of the combustion simulation. The location where the flame goes from blue to bright red in a relatively short distance is the location of the highest temperature gradients. Consequently, figure 41 shows that the highest negative numbers seem to be occurring in and around the regions where the temperature gradient is large.

For all locations in the domain the fourth term has negative values. Away from the flame the negative values are much lower, due predominately to little or no temperature gradient. This signifies that the entropy which is generated away from the flame is less than the entropy generated close to the flame for the fourth term of the entropy inequality.

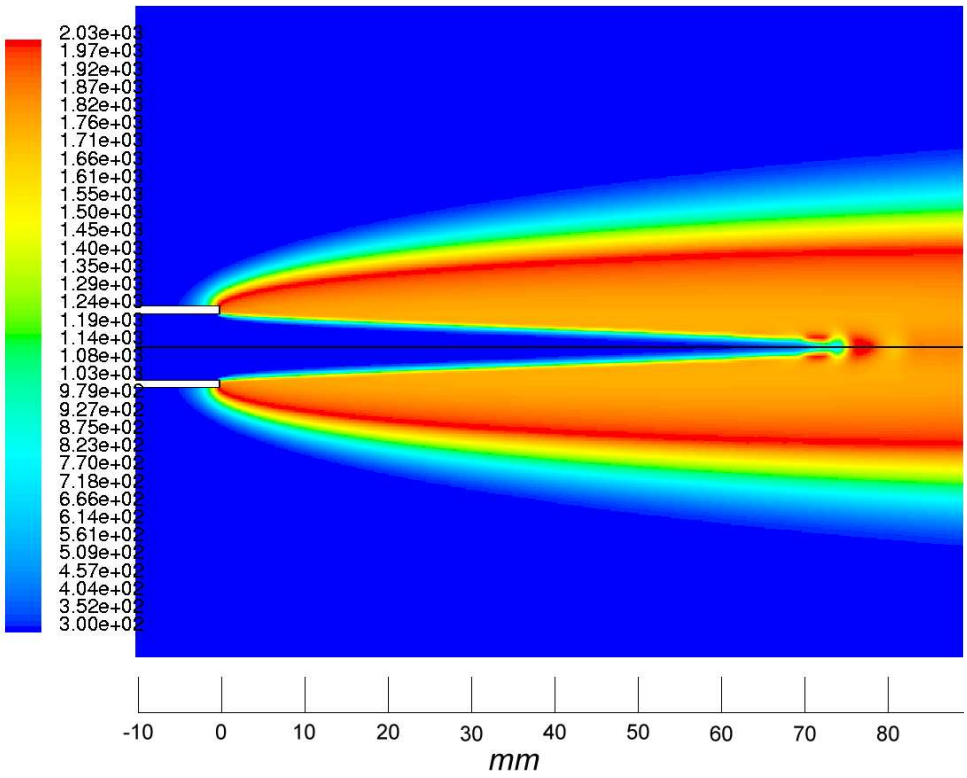


Fig. 40. Contour plot showing temperature variation of Sandia simulation.

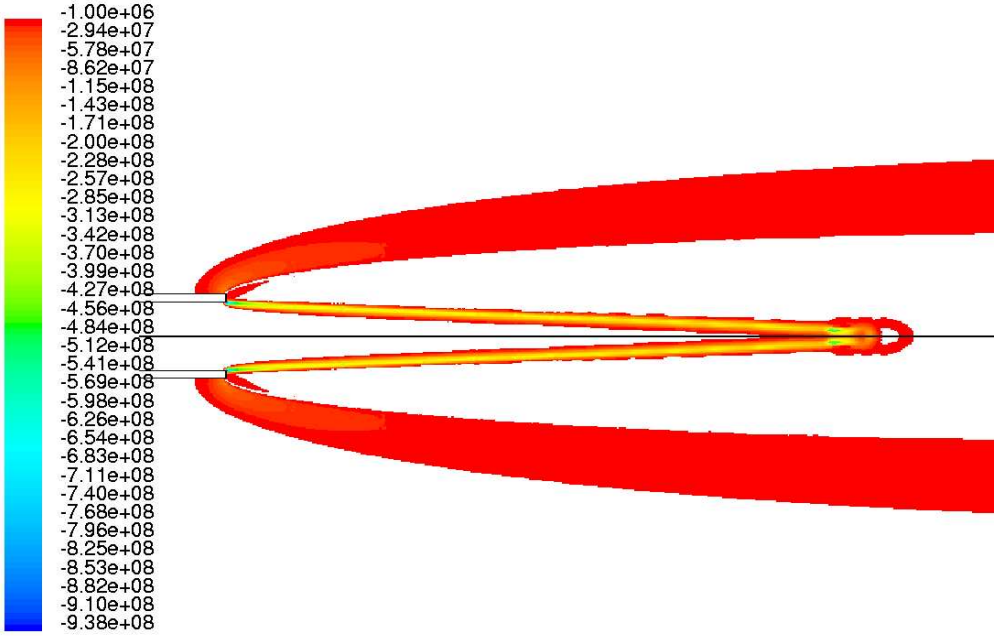


Fig. 41. Contour plot showing values of fourth term of entropy inequality.

e. Overall Inequality Satisfaction

Thus far the discussion has been limited to each of the four terms independently. What happens when these terms are summed together? Is the entropy inequality satisfied? This section will show the results obtained when the overall inequality is examined.

As previously mentioned, different numerical simulations were performed. The only difference between the simulations is the value of the temperature limiter used. After creating the numerical spark, the problem was run for many iterations keeping the limiting temperature at $2025K$. After a converged solution had been obtained, the solution was saved. This solution was then called the starting solution. From this starting solution five different cases were run, each corresponding to a different temperature limiter. Four of the cases had temperature limiters of $2300K$, $2600K$, $2900K$ and $5000K$. The fifth case continued to advance the solution holding the temperature limit at $2025K$.

The objective was to obtain converged solutions and look at the effects the different solutions had on the second law of thermodynamics. It was found that all cases had certain cells where the entropy inequality was violated. However, different cases had different numbers of points which violated the second law. If q is the number of points within the domain that violate the second law then figure 42 shows $\log_{10}(q)$ versus the number of iterations.

For the three cases where the temperature limiter was the lowest, $2025K$, $2300K$ and $2600K$, the number of points which violated the second law oscillated. However, there seemed to be an overall slight decrease as the number of iterations increased, and typically, the simulation with the lowest temperature limiter had the fewest number of cells within the domain in violation of the entropy inequality. For the two cases with

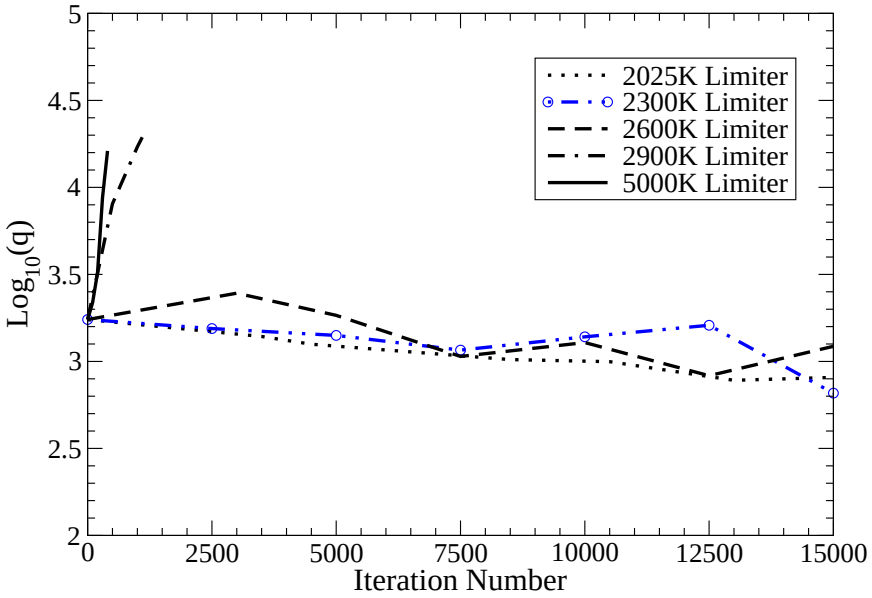


Fig. 42. Comparison of the advancement of the solution and the number of points which violated the second law.

the high temperature limiters, $2900K$ and $5000K$, the number of points that violated the second law increased exponentially. The case with $5000K$ temperature limiter diverged in less than 500 iterations. The case with $2900K$ limiter, started yielding solutions which looked nothing like the experimental data after 1000 iterations. This was because the location where combustion was occurring in the domain moved from the exit of the tube, to down inside the tube.

In addition to counting the number of cells where the entropy inequality is violated, it is also desired to explore the magnitude of the entropy inequality at each point of violation. When each term in the entropy inequality was examined independently, it was discovered that the terms which had high magnitude negative numbers were terms where more disorder was likely to occur. Therefore, the magnitude of the inequality seemed to have physical significance. This trend should continue for cells which violate the second law of thermodynamics as well. This means having high magnitude positive values for the entropy inequality is worse than having low mag-

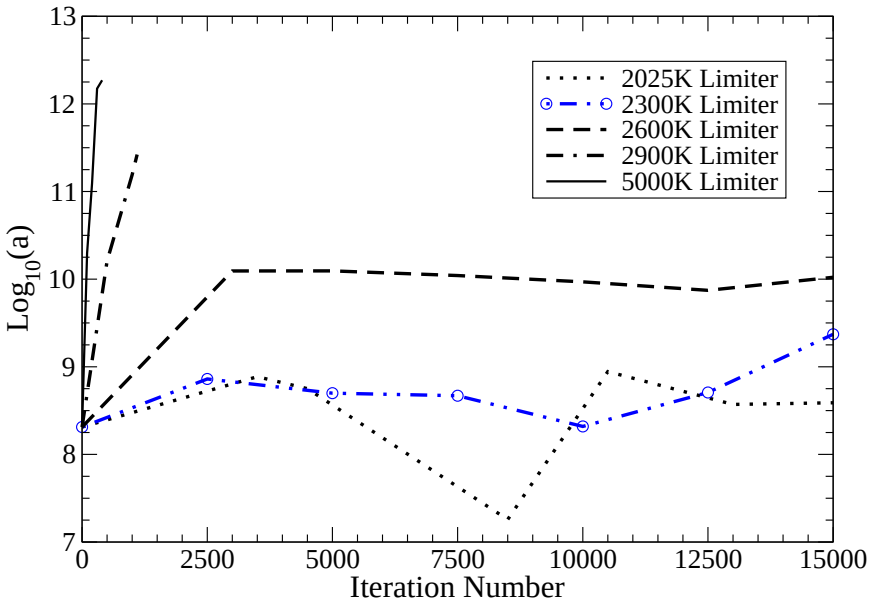


Fig. 43. Comparison of the advancement of the solution and the magnitude of the entropy violation.

nitude positive values. Therefore, to explore the effects of the magnitude of entropy inequality, a summation over all points in violation was performed. The variable a is the result from the calculation of the entropy inequality, equation (6.5), after it is summed over all points which were in violation. Mathematically it may be written as

$$\begin{aligned} a &= \sum_{j=1}^{N_v} w \\ w &= (1^{st}term + 2^{nd}term + 3^{rd}term + 4^{th}.term) \end{aligned} \tag{6.58}$$

N_v is the number of points in violation, and w is the calculation of the entropy expression at a cell. The result is a number with a certain positive magnitude, a. Figure 43 shows the variation of the base 10 log of the magnitude, a, versus the iteration number for each temperature limiter.

The two cases with upper temperature limits show large increases in the magnitude of the summation, a, as the simulation is advanced. Similar results were shown with the number of points that violate the second law. Also, the two cases with the

highest temperature limit provide the worst comparison with the experimental data. For the case with the $5000K$ limiter, the solution diverges after 490 iterations. The case with $2900K$ limiter starts yielding unphysical solutions after 1000 iterations. Another general trend is that the higher the temperature limit, the higher the value of the sum over all of the points which violate the entropy inequality. This suggests that the magnitude, a, effects the solution obtained, and that generally, the lower the magnitude the closer the solution is to the experiment.

The next stage investigates where inside the domain the second law of thermodynamics is violated. Figures 44-47 show three-dimensional plots. On the $z = 0$ plane, a red line shows the boundary of the numerical domain. The x-axis shows the axial location and the y-axis denotes the radial direction in m. Remember the x-axis is the center line or axisymmetric axis. The z-axis plots the $\log_{10}$ of the number calculated from the entropy inequality, i.e. w from equation (6.58). The only points which are plotted are the points which violate the entropy inequality, that is where the entropy inequality has a positive value.

In general the points that violate the second law occur along the axisymmetric axis. This is because the flame is located along this axis. Another interesting effect is that the closer the points are to the $x = 0$ axial location, the higher positive number the entropy inequality has. In all cases, the magnitude seems to decrease as the axial location increases.

The number of cells that violate the second law of thermodynamics and the number of cells that are actually limited by the temperature limit are not directly correlated. Rather, the temperature limit seems to bound the way the solution behaves. As the temperature limit is increased, the simulation constraints are relaxed, and the numerical simulation has more freedom to define itself. What is actually seen is as the temperature limit is raised, the solution becomes unstable and even diverges.

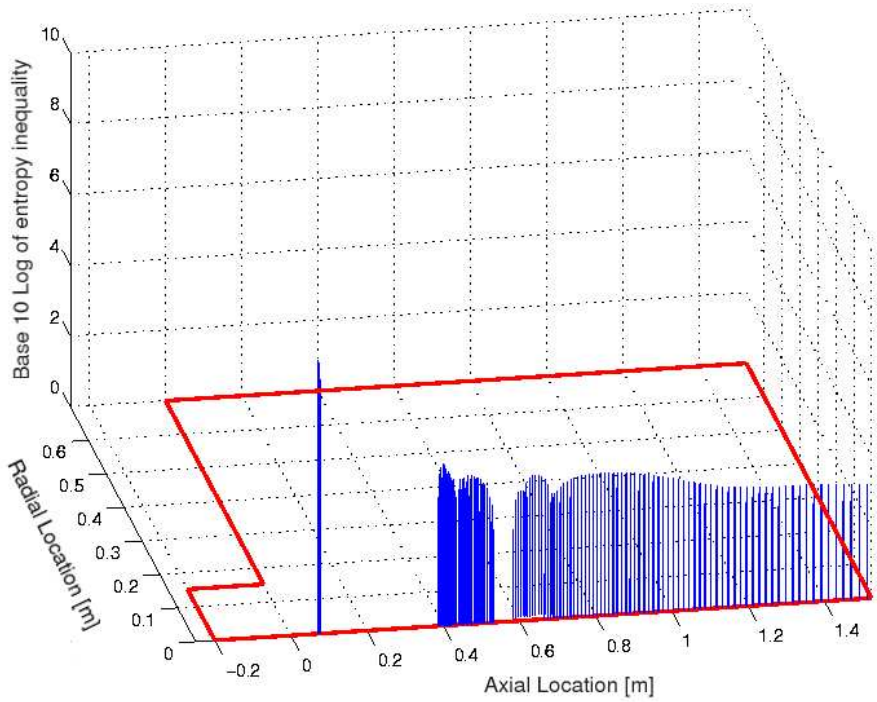


Fig. 44. Locations where entropy inequality is not satisfied for temperature limiter of $2025K$ at 15000 iterations.

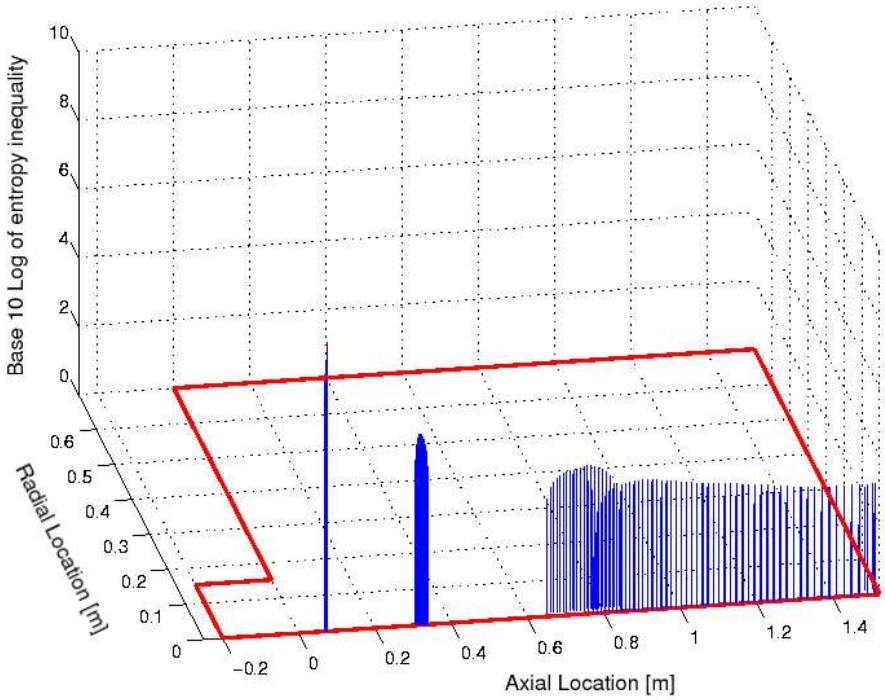


Fig. 45. Locations where entropy inequality is not satisfied for temperature limiter of $2300K$ at 15000 iterations.

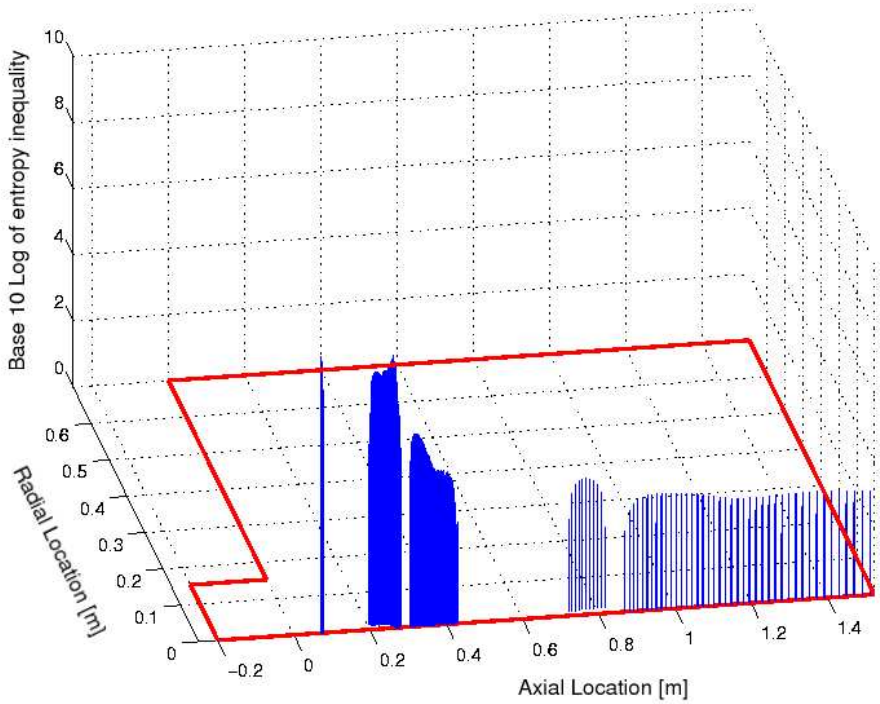


Fig. 46. Locations where entropy inequality is not satisfied for temperature limiter of $2600K$ at 15000 iterations.

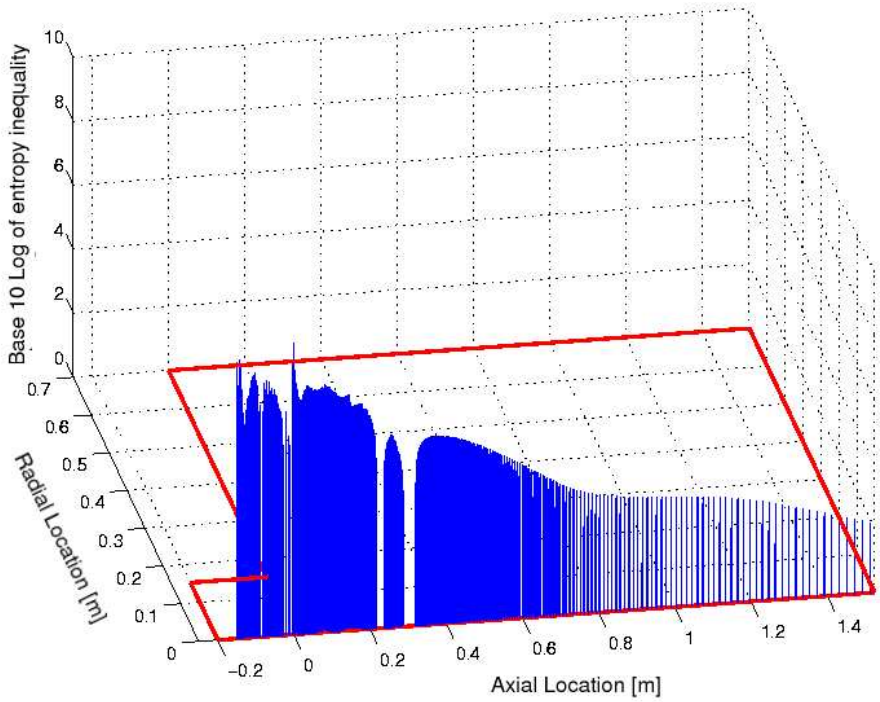


Fig. 47. Locations where entropy inequality is not satisfied for temperature limiter of $2900K$ at 1000 iterations.

This is clearly seen in figures 44-47. Figure 46 has more points with high magnitude values than figure 44. Figure 47 shows even more points with high magnitudes than figure 46, and at this moment in the simulation, the data from figure 47 does not compare well with experimental data. So it is indeed seen that as the temperature limit increases, progressively more cells inside the domain have a high positive value for the entropy inequality calculation.

Therefore it is necessary to place some manner of constraint on the solution. The temperature limiter provides a constraining mechanism, however, the physical reason for using it is unclear. This seems an *ad hoc* method for constraining the simulation. A more physically significant constraint on the numerical simulation is to enforce that the second law of thermodynamics be satisfied at every point. It is known that this physical law must be satisfied, so it should be necessary that a numerical simulation satisfy this law at every point within the domain during every iteration.

F. Conclusions and Future Work

It has been shown for this particular numerical simulation of combustion that the second law of thermodynamics is violated. A temperature limiter has been enforced to limit how much the second law is violated and to allow for a reasonable converged solution. It has been proposed that instead of a temperature limiter, which has a vague physical basis, that the second law inequality be used as a limiter. Further work is needed in order to determine how the entropy inequality can be used as a limiter in combustion simulations.

The combustion model used herein may not be the most sophisticated simulation available, but it is still believed to be a viable way of computing a combustion simulation. The entropy inequality should be checked for other simulations, which use other

combustion models, in order to help define which models are best. The next step in this research would be to test the entropy inequality for reaction mechanisms that include more chemical reactions. Other reduced mechanisms exist for methane/air flames. The next step would be to include a reduced mechanism that includes a three-step reaction mechanism and then move on to a six-step mechanism.[23,25]

In order to look more closely at the combustion models, the likely culprit of the entropy violation, it is necessary to perform direct numerical simulation. This would allow the Navier-Stokes equations to be solved with no model approximations and would place the focus squarely on the combustion model. This is the best way to examine the different combustion models to decide when models are physically realistic.

CHAPTER VII

CONCLUSIONS

Computational fluid dynamics (CFD) has been used to simulate combustion flows and flows with moving boundaries. This chapter will focus on a brief summary of many of the results from this research. It will begin by describing the physical and numerical model used in the associated simulations and will be followed by summaries of the results of the moving and deforming mesh and combustion simulations.

A. Discussion of Physical Model and Numerical Implementation

The physical equations that govern a fluid dynamics simulation and a combustion simulation have been introduced. These physical transport equations have many important terms that need further explanation. Some of the expressions for the most important terms for this research have been presented. For a combustion simulation this includes the presentation of the species balance equations as well as expressions used for the rate of reactions, constant pressure specific heat, dynamic viscosity and diffusion of individual species. For problems dealing with the moving and deforming boundaries no additional physical equations are necessary.

Next a brief outline of the numerical model used to solve the physical equations was given. This discussion was intentionally left brief as it was not the focus of the current research. However, it is important enough that some discussion was devoted as to how the equations were solved and what assumptions were used in the numerical model. All simulations were done using a segregated, finite-volume, implicit numerical algorithm. Detailed information can be found by looking in the FLUENT users manual[4] or by looking at a selected computational fluid dynamics text.[3] For combustion simulations the segregated numerical flow solver adds transport equations

for individual species that are being modeled. These transport equations are then solved for the species mass fraction of all the species in each of the cells within the domain. The species mass fraction is a scalar quantity, so the solution of the mass fraction uses the same procedures used when solving other scalar transport quantities, such as energy or turbulence quantities.

For moving and deforming mesh problems it is necessary to use an unsteady simulation method because the boundaries of the domain, or the flow behavior within the domain, change with time. In addition to the unsteady simulation, numerical techniques had to be added to restructure the mesh to allow for the movement of a rigid body. This was done in two different ways, each with its own advantages and disadvantages. The first method modeled grid deformation using a spring analogy. This method is computationally efficient because it does not change the connectivity information of the mesh, however it does not always allow for large deformations. The second method used to restructure a mesh is called the local remeshing method. It works by taking a collection of cells whose skewness has become too large and removing them. It then creates entirely new cells inside of this region. Its main advantage is that it works for large deformations but it is generally more computationally expensive than the spring method. Also, it can become quite difficult to have absolute control over the new cells when remeshing is performed.

B. Investigation into Moving Rigid Boundaries in CFD

An important step in an aeroelastic calculation is obtaining a flow solution around an object. However, because aeroelastic problems have moving and deforming boundaries, it is necessary to include a moving and deforming algorithm in addition to a standard stationary flow solver. The commercially available FLUENT fluid dynam-

ics flow solver was used because of its ability to alter a mesh as a flow solution is solved. A FLUENT user defined function (UDF) was used to set the angular and translational velocities of a rigid body.

Several tests were performed to investigate the simulation of unsteady aerodynamics with a moving deforming mesh. The first test assigned a vertical sinusoidal movement to a circular cylinder. The next test evaluated FLUENT's unsteady solver algorithm. The flow around a stationary circular cylinder was calculated and the shedding frequency of the vorticies from the cylinder was matched with experimental data. The final test was to use a UDF to assign a wing the flapping motion of a forward flying hornet and solve for the flow around the moving wing. This successful simulation lays the ground work for more advanced simulations with rigid boundaries and even deformable bodies. Simulations like these will also foster further development of the algorithm necessary to study fully coupled fluid-structure interaction problems. Computer simulations capable of encompassing aeroelastic interactions will become the future design tool for aircraft and will be essential in development of micro aerial vehicles (MAVs) that fly just as insects do.

C. Validation of Simple Combustion Model for *In Situ* Reheat Investigation

Thermodynamic analysis has shown an increase in power generation of a turbine due to a method called *in situ* reheat. A combustion CFD analysis allows for initial examination of *in situ* reheat within a turbine without costly experimental testing. Many different simulations must be performed to test a variety of different test configurations of fuel injection into a turbine. Before a computer simulation is performed it is necessary to find an efficient chemical model and validate its use. The chemical model must not be too complicated and computationally inexpensive, while at

the same time provide accurate temperature calculations. Two different combustion models were tested against a single vane burner experiment. This experiment was designed to mimic flow conditions of the first stator in a gas turbine. The numerical simulation of the single vane burner validated the use of a two step finite rate methane/air combustion model which included a reversible carbon monoxide reaction. The numerical simulation matched temperature and carbon monoxide levels as close as 1.2% and 9% respectively. The chemical model validated in this research was then used in an actual *in situ* reheat simulation. This simulation showed a 2.8% to 5.1% increase in power generation when compared to a simulation of the same turbine without *in situ* reheat.[19]

D. Further Investigation of Simple Combustion Model and Its Ability to Satisfy the Second Law of Thermodynamics

The next step involved further investigation into simple combustion models. Another simulation was devised to test how well a simple combustion model could match laminar flame results from Sandia National Laboratories. A laminar flame removes the complications of turbulence modeling and allows more attention to be paid to the combustion model. Another method of evaluating a combustion model is obtained through the second law of thermodynamics. The type of combustion models used in this research are created empirically by curve fitting parameters such that a simulation matches experimental data. However, no requirement is in place that forces the combustion model to satisfy the second law of thermodynamics. Therefore, when one of these combustion models is used within a combustion simulation, there is no guarantee that the second law is automatically satisfied. A simulation was created to examine whether or not the second law is satisfied when using a sim-

ple combustion model to simulate a laminar flame. Initially the simulations would diverge. This required the use of a temperature limiter. The limiter set an absolute maximum on what the highest stored temperature value could be at a given cell. Five different temperature limiters were used with values of 2025, 2300, 2600, 2900, and $5000K$. Simulations using different temperature limiters provided different numerical solutions. A comparison between the experimental data and all numerical solutions showed that all of the simulations violated the second law of thermodynamics. Interestingly, a general trend showed that as the temperature limiter was increased, the second law was violated more. When the simulation with a temperature limiter of $5000K$ diverged after 490 iterations, it contained 12,000 cells in violation of the second law. In contrast the simulation with a temperature limiter of $2025K$ did not diverge and after 15,000 iterations had only 1,397 cells in violation. Temperature limits of 2025, 2300, and $2600K$ resulted in converged solutions.

Another interesting aspect this analysis is not just how many points are in violation but also magnitude of the violation. Cells that violate the second law with high magnitudes occurred frequently and in large amounts before unphysical solutions and divergence. The numerical simulation should converge to a single solution no matter what the value of the temperature limiter. The temperature limiter provides an unphysical boundary that keeps the solution from diverging. However, the second law of thermodynamics provides an important physical constraint that should be used in place of the temperature limiter.

E. Further Applications

Each of the topics discussed in the research presented can easily become their own interesting and fully independent research topic. Much more work is needed in the

development of fully coupled aeroelastic computer simulation. Even with this moving and deforming model for a rigid body, more is needed for a fully coupled fluid-structure interaction problem. The next step in this research would be to perform a fluid-structure interaction for an elastically mounted cylinder. An elastically mounted cylinder is a dynamical system that would allow the cylinder to move according to the unsteady loads resulting from the shedding of the vorticies. This would prove that a fluid-structure interaction of a rigid body could be performed with FLUENT. The next extension would be to perform tests for a deformation of the body. However, deformable bodies may require the use of a structural model to be added to the FLUENT simulation.

In situ reheat is an old idea that, with the advent of more advanced computer simulations, may prove to be a valuable way of improving performance of turbomachinery equipment. Because the numerical simulations show such great benefits in power when using *in situ* reheat, further work should be done to verify this. This could include simulations with more complex combustion models. After that a full three-dimensional simulation should be performed to take into account the radial variation of the turbine. More advanced and successful simulations can then lead to the actual experimental testing of *in situ* reheat.

As many different theories are brought together and used to form the next level of computational fluid dynamics, it is necessary to continue to check that these simulations satisfy the governing laws of physics. This research only scratches the surface of an investigation into combustion simulations. However, the importance of requiring that the second law of thermodynamics be satisfied at every location at every moment of a combustion simulation cannot be overstated. If the simulation is providing physically unrealistic solutions, how can it be used to model a physical problem? The findings in this research are very important and should be continued. The next step

would be to use more advanced combustion models. Instead of modeling only two of the reactions taking place, more should be modeled to see if additional reactions will satisfy the second law of thermodynamics. In addition to investigating combustion models, the second law of thermodynamics should be used as a limiting parameter during the simulation process. This will allow physically realistic simulations even when using simple combustion models which themselves may not satisfy the second law. If these additional steps are investigated, a more complete understanding of how to model combustion will result.

REFERENCES

[1]Sirignano, W. A., and Liu, F., "Performance Increases for Gas-Turbine Engines Through Combustion Inside the Turbine," *Journal of Propulsion and Power*, Vol. 15, No. 1, 1999, pp.111-118.

[2]Bisplinghoff, R. L., Ashley, H., and Halfman, R. L., *Aeroelasticity*, Dover, New York, 1996, p.1.

[3]Tannehill, J. C., Anderson, D. A., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, 2nd Ed., Taylor & Francis, Philadelphia, PA, 1997, pp.249-350.

[4]Fluent Inc., *Fluent 6.1: Users Guide*, Lebanon, NH, January 2004.

[5]Reid, R. C., Prausnitz, J. M., and Poling, B. E., *The Properties of Gases and Liquids*, 4th Ed., McGraw-Hill Inc., New York, 1987, pp.29-33,392-393,733-734.

[6]Kuo, K. K., *Principles of Combustion*, Wiley Inc., New York, 1986, pp.31-39,109-163.

[7]McBride, B. J., Gordon, S., and Reno, M. A., "Coefficients for Calculating Thermodynamic and Transport Properties of Individual Species", NASA Technical Memorandum 4513, Washington, DC, October 1993.

[8]Slattery, J. C., *Advanced Transport Phenomena*, Cambridge University Press, Cambridge, UK, 1999, pp.41,436-450,451-465.

[9]Merk, H. J., "The Macroscopic Equations for Simultaneous Heat and Mass Transfer in Isotropic, Continuous and Closed Systems", *Appl. Sci. Res.*, Section A, Vol. 8, 1958, pp.73-99.

[10]Issa, R. I., "Solution of the Implicitly Discretized Fluid Flow Equations by Operator-Splitting," *J. Comput. Phys.,* Vol. 62, 1985, pp.40-65.

[11]Fluent Inc., *Fluent User Defined Functions Manual*, Lebanon, NH, January 2004.

[12]Anderson, J. D., *Fundamentals of Aerodynamics*, 2nd Ed., McGraw-Hill Inc., New York, 1991, pp.228-236.

[13]Alonso, J., Martinelli, L., and Jamseson, A., "Multigrid Unsteady Navier-Stokes Calculations with Aeroelastic Applications", AIAA Paper 95-0048, January 1995.

[14]Blackburn, H. M., and Karniadakis, G. E., "Two- and Three-Dimensional Simulations of Vortex Induced Vibration of a Circular Cylinder", Isope-93 Conference, Singapore, 1993.

[15]Dickinson, M. H., Lehmann, F. O., and Sane, S. P., "Wing Rotation and the Aerodynamic Basis of Insect Flight" *Science Magazine*, Vol. 284, June 18, 1999, pp.1954-1960.

[16]Sane, S. P., Dickinson, M., Nakahashi, K., and Kato, T., "Flow Simulation of Flapping Wings of an Insect Using Overset Unstructured Grid", AIAA Paper A01-31162, June 2002.

[17]Morton, S. A., Melville, R. B., and Visbal, M. R. "Accuracy and Coupling Issues of Aeroelastic Navier-Stokes Solutions on Deforming Meshes", AIAA Paper 97-1085, April 1997.

[18]Westbrook, C., and Dryer, F. L., "Simplified Reaction Mechanisms for the Oxidation of Hydrocarbon Fuels in Flames" *Combustion Science and Technology*, Vol. 27, 1981, pp.31-43.

[19]Chambers, S. B., Flitan, H. C., Cizmas P. G. A., Bachovchin, D., Lippert, T., and Little, D., "The Influence of In Situ Reheat on Turbine-Combustor Performance" *The 49th ASME International Gas Turbine Congress*, Vienna, Austria, June 2004.

[20]Isvoranu, D D., and Cizmas, P. G. A., "Numerical Simulation of Combustion and Rotor-Stator Interaction in a Turbine Combustor", *International Journal of Rotating Machinery*, Vol. 9, 2003, pp.363-374.

[21]Flitan, H. C., and Cizmas, P. G. A., "Analysis of Unsteady Aerothermodynamics Effects in a Turbine Combustor", *The 10th International Symposium on Unsteady Aerodynamics and Aeroelasticity of Turbomachines*, Durham, North Carolina, September 2003.

[22]Cizmas, P. G. A., Flitan, H. C., and Isvoranu, D. D., "Numerical Prediction of Unsteady Blade Loading in a Turbine Combustor", *8th National Turbine High Cycle Fatigue Conference*, Monterey, CA, April 2003.

[23]Peters N., and Williams, F. A., "The Asymptotic Structure of Stoichiometric Methane-Air Flames", *Combustion and Flame*, Vol. 68, 1987, pp.185-207.

[24]Nicol, D. G., Malte, P. C., Hamer, A. J., and Roby, R. J., "Development of a Five-Step Global Methane Oxidation - NO Formation Mechanism for Lean-Premixed Gas Turbine Combustion", *Transactions of the ASME, Journal of Engineering for Gas Turbines and Power*, Vol.121, 1999, pp. 272-280.

[25]Seshadri, K., Bai, X. S., and Pitsch H., "Asymptotic Structure of Rich Methane-air Flames", *Combustion and Flame*, Vol. 127, 2001, pp.2265-2277.

[26]Jones, W. P., and Lindstedt, R. P., "Global Reaction Schemes for Hydrocarbon Combustion", *Combustion and Flame*, Vol. 73, 1988, pp.233-249.

[27]Personal communication from A. N. Karpetis and R. S. Barlow, Sandia National Laboratories, Livermore, CA.

[28]Taylor, R., and Krishna, R., *Multicomponent Mass Transfer*, John Wiley & Sons Inc., New York, 1993, pp.80-88.

[29]Bird, R. B., Stewart, W. E., and Lightfoot, E. N., *Transport Phenomena*, John

Wiley & Sons Inc., New York, 1960, pp.511,554-590.

[30]Smith, J. M., and Van Ness, H.C., *Introduction to Chemical Engineering Thermodynamics*, McGraw-Hill Inc., New York, 1987, pp.504-516.

APPENDIX A

SAMPLE UDF FOR CALCULATION OF SECOND LAW OF THERMODYNAMICS

```
/* UDF which brings together all of the parts and computes */
/* the entropy inequality, eq (8.4.3-8) pg 451, "Advanced */
/* Transport Phenomena"  John C Slattery     */

/* AUTHOR: Steven Chambers */
/* Date:   May 10, 2004  */

#include <stdio.h>
#include "udf.h"
#include <math.h>
const char FILE_NAME[] = "output.txt";
const char FILE_NAME1[] = "tempor.txt";
const char FILE_NAME2[] = "entr.txt";

DEFINE_ON_DEMAND(summation)
{

  Domain *d; /* declare domain pointer since it is not passed as an */
  /* argument to the DEFINE macro  */
  long int i, j, k, iamintrouble;
  FILE *out_file; // output file
  FILE *tem_file; // temporary output file
  FILE *ent_file; // Output file which counts the number of points which
                  // are in violation

  static double lnkreac1, lnkreac2, lnkreac3;
  static double pro, defaul;
  static double itemp, tempsq, itempsq;
  static double r11, r12, r13, r14, r15, r16;
  static double r21, r22, r23, r24, r25, r26;
  static double Ar[3], Mm[6], Ea[3];
  static double stoicp[3][6],stoicpp[3][6],rateexp[3][6],rateexpp[3][6];
  static double sdensity[6], density, Cm[6], sc[3][6];
  static double Y[6],X[6],N[6],kf[3],rhr[3],rr[3][6];
  static double fin1[6], fin2[6], fin3[6], fina, final, k1,k2,k3,uni;
  static double xloc[3], yloc;
```

```
static double kl;

static double sigmar[6][6],epokr[6][6],td_star[6][6];
static double omega[6][6],ooMm[6],numer,denom,diffcoef[6][6];
static double thermcoef,numsum,densum,middle[6],tdiffcoef[6];
static double Mmm, xx[5][5], ans1[5], ans2[5],one,two,three;

static double DA[5][5], DB[5][5], aii1, aii2, aij1, aij2, bii1,bii2;

long int imax, imax2, np, ii, ii2, ll, ll2;
static double yin[5][5],DAO[5][5],ide[5][5],xxo[5][5],ud,ud2;
static double indx[30],indx2[30], aamax, aamax2, dum, dum2;
static double sum, sumx, sum2, vv[100], vv2[100];
static double yin2[5][5], ide2[5][5], bb[5], bb2[5];
static double DIJ[5][5];

static double A,B,C,D,E,F,tempe,Pabs,sigma[6],epok[6];
static double gradtx,gradty,gradYx[5],gradYy[5];
static double ctotal, ntotal,X2[6];

static double mgrx[6], mgry[6], sumy, vecJx[6], vecJy[6];
static double secondterm, dot, dotx, doty;
static double xpart, ypart, fourth, epsilonx, epsilony;

long int ident[2][2];
static double gradv[2][2], gradvt[2][2], tau[2][2], first[2][2];
static double delv, tempo[2][2], end, piece, mu;
static double entropy;
static double ymin;  /* cut-off value below which mass
                        fraction is zero   */
static char yzero[6];/* flag that indicates whether mass
                        fraction is zero */
static char xzero[6];/* flag that indicates whether mole
                        fraction is zero */
/* If mole fraction or mass fraction is below 10^(-9) then
   the portion of that particular species is assumed to be small
   and therefore not included in summation */
static double term1, term2;/* temporary terms in summations       */
static double summation; /* adds up all entropy violation values  */

Thread *t;
cell_t c;
d = Get_Domain(1);   /* Get the domain using Fluent utility */
```

```
/* Defines coefficients for the equilibrium
   constants as function of temp */
r11 = 62993;
r12 = 3.155;
r13 = -3.1915e-03;
r14 = 0.36066e-06;
r15 = 0.2757e05;
r16 = -9.25;

r21 = 34465.05;
r22 = 0.2615;
r23 = 0.1175e-03;
r24 = 0.0;
r25 = -0.50625e05;
r26 = -12.80;
//---------------------------------------------------------------------

/*  Ar, Arrhenuis Coefficients;  Ea, Activation Energy ---------------*/
Ar[0] = 5.012e11;
Ar[1] = 2.239e12;
Ar[2] = 5.0e08;

Ea[0] = 2.0e08;
Ea[1] = 1.7e08;
Ea[2] = 1.7e08;
//---------------------------------------------------------------------

Mm[0] = 16.04303; /* CH4 [0]*/
Mm[1] = 31.9988;  /* O2  [1]*/
Mm[2] = 44.00995; /* CO2 [2]*/
Mm[3] = 28.01055; /* CO  [3]*/
Mm[4] = 18.01534; /* H2O [4]*/
Mm[5] = 28.0134;  /* N2  [5]*/

out_file = fopen(FILE_NAME, "w");
tem_file = fopen(FILE_NAME1, "w");
ent_file = fopen(FILE_NAME2, "w");
if (out_file == NULL){
  printf("Cannot opent %s\n", FILE_NAME);
  exit(8);
}

for (i=0;i<3;i++){
  for (j=0;j<6;j++){
    stoicp[i][j] =0.0;
```

```
    stoicpp[i][j]=0.0;
    rateexp[i][j]=0.0;
    rateexpp[i][j]=0.0;
    sc[i][j] = 0.0;
  }
}
stoicp[0][0]  = 1.0; // Assigns stoichiometric coefficent for methane
                     // in reaction 1
stoicp[0][1]  = 1.5; // Assigns stoichiometric coefficent for oxygen
                     // in reaction 1
stoicpp[0][3] = 1.0; // Assigns stoichiometric reactant coefficient
                     // for CO in reac1
stoicpp[0][4] = 2.0;

stoicp[1][3]  = 1.0;
stoicp[1][1]  = 0.5;
stoicpp[1][2] = 1.0;

stoicp[2][2]  = 1.0;
stoicpp[2][3] = 1.0;
stoicpp[2][1] = 0.5;

sc[0][0]      = -1*stoicp[0][0];
sc[0][1]      = -1*stoicp[0][1];
sc[0][3]      = stoicpp[0][3];
sc[0][4]      = stoicpp[0][4];
sc[1][3]      = -1*stoicp[1][3];
sc[1][1]      = -1*stoicp[1][1];
sc[1][2]      = stoicpp[1][2];
sc[2][2]      = -1*stoicp[2][2];
sc[2][3]      = stoicpp[2][3];
sc[2][1]      = stoicpp[2][1];

rateexp[0][0] = 0.7;
rateexp[0][1] = 0.8;

rateexp[1][3] = 1.0;
rateexp[1][1] = 0.25;
rateexp[1][4] = 0.5;

rateexp[2][2] = 1.0;


sigma[0] = 3.758;  /* CH4 */
sigma[1] = 3.467;  /* O2  */
```

```
  sigma[2] = 3.941;  /* CO2 */
  sigma[3] = 3.69;   /* CO  */
  sigma[4] = 2.641;  /* H2O */
  sigma[5] = 3.798;  /* N2  */

  epok[0]  = 148.6;
  epok[1]  = 106.7;
  epok[2]  = 195.2;
  epok[3]  = 91.7;
  epok[4]  = 809.1;
  epok[5]  = 71.4;

  A = 1.16145;  //
  B = 0.14874;  //
  C = 0.52487;  //  These coeffcients come from pg 393
  D = 0.77320;  //   "Properties of Gases & Liquids"
  E = 2.16178;  //    Reid, Prausnitz, Poling
  F = 2.43787;  //Also, sigma and epok (e/k) come from same source p 733.
  uni = UNIVERSAL_GAS_CONSTANT;

  defaul = -1.0e+10;
  ymin   = 0.0000001;
  summation = 0.0;

  /* Loop over all cell threads in the domain */
  thread_loop_c(t,d)
    {
      /* Loop over all cells */
      begin_c_loop(c,t)
{

  //---------Get Flow properties from cells-----------------
  density = C_R(c,t);
  tempe = C_T(c,t);
  Pabs  = 101325 + C_P(c,t);
  for (i=0;i<6;i++){
    Y[i] = C_YI(c,t,i);
  }
  gradtx = C_T_G(c,t)[0];
  gradty = C_T_G(c,t)[1];
  for (i=0;i<5;i++){
    gradYx[i] = C_YI_G(c,t,i)[0];
    gradYy[i] = C_YI_G(c,t,i)[1];
  }
  kl = C_K_L(c,t);
```

```
  mu = C_MU_L(c,t);
  //-------------------------------------------------------

  //----------------Calculate mole fraction-----------------------
  ctotal = 0.0;
  ntotal = 0.0;
  for (j=0;j<6;j++){
            yzero[j] = 'f';
    if (Y[j]<0.01 * ymin){
/* If mole fraction or mass fraction is below 10^(-9) then
     the portion of that particular species is assumed to be small
     and therefore not included in summation */
              yzero[j] = 't';
    }
    sdensity[j] = Y[j]*density;
    Cm[j] = sdensity[j]/Mm[j]; /*Calculation of molar concentration*/
    N[j] = 100*Y[j]/Mm[j];
    ntotal = ntotal + N[j];
    ctotal = ctotal + Cm[j];
  }
  for (j=0;j<6;j++){
    xzero[j] = 'f';
    X[j]=N[j]/ntotal;/* Calculation of mole fraction */
    X2[j]=Cm[j]/ctotal;
    if (X[j] < ymin * 0.01){
/* If mole fraction or mass fraction is below 10^(-9) then
     the portion of that particular species is assumed to be small
     and therefore not included in summation */
      xzero[j] = 't';
    }
  }
  //-------------------------------------------------------------

  //----------------Calculate Arrhenius Rate of Reaction------------
  itemp = 1/tempe;
  tempsq = tempe*tempe;
  itempsq = 1/tempsq;
  lnkreac1 = r11*itemp+r12*log(tempe)+r13*tempe+\
                     r14*tempsq+r15*itempsq+r16;
  lnkreac2 = r21*itemp+r22*log(tempe)+r23*tempe+\
                     r24*tempsq+r25*itempsq+r26;
  lnkreac3 = -1*lnkreac2;
  fprintf(out_file,"lnk1 %f\n",lnkreac1);
  k1=exp(lnkreac1);
  k2=exp(lnkreac2);
```

```
  k3=1.0/k2;

  for (i=0;i<3;i++){
     kf[i]=Ar[i]*exp((-1*Ea[i])/(UNIVERSAL_GAS_CONSTANT*tempe));
   }

  for (i=0;i<3;i++){
    pro = 1;
    for (j=0;j<6;j++){
      pro = pro * pow(Cm[j],rateexp[i][j]);
    }
    rhr[i] = kf[i] * pro;
  }
  //---------------------------------------------------------------------
  for (j=0;j<3;j++){
    C_UDMI(c,t,j) = rhr[j];
  }
  for (i=0;i<3;i++){
    for (j=0;j<6;j++){
      rr[i][j] = Mm[j]*(stoicpp[i][j]-stoicp[i][j])*rhr[i];
    }
  }
  fina = 0.0;
  fprintf(out_file,"rr %f\n",rr[0][1]);
  fprintf(out_file,"k1 %f\n",k1);
  for (j=0;j<6;j++){
/* If mole fraction or mass fraction is below 10^(-9) then
     the portion of that particular species is assumed to be small
     and therefore not included in summation */
    if (xzero[j] == 't'){
      fin1[j] = 0.0;
    }
    else{
      fin1[j] = log((1.0/k1)*pow(1.0*X[j],sc[0][j]))*rr[0][1]/\
                        (Mm[1]*(-1)*stoicp[0][1]);
    }
    fina = fin1[j] + fina;
  }
  for (j=0;j<6;j++){
    if (xzero[j] == 't'){
      fin2[j] = 0.0;
    }
    else{
      fin2[j] = log((1.0/k2)*pow(1.0*X[j],sc[1][j]))*rr[1][1]/\
                        (Mm[1]*(-1)*stoicp[1][1]);
```

```
  }
  fina = fin2[j] + fina;
}
for (j=0;j<6;j++){
  if (xzero[j] == 't'){
    fin3[j] = 0.0;
  }
  else{
    fin3[j] = log((1.0/k3)*pow(1.0*X[j],sc[2][j]))*rr[2][1]/\
                      (Mm[1]*stoicpp[2][1]);
  }
  fina = fin3[j] + fina;
}
final = UNIVERSAL_GAS_CONSTANT*tempe*fina;
C_UDMI(c,t,3) = final;
fprintf(out_file,"Something %g\n", final);
if (final >= 0.0){
  /* xloc = C_CENTROID(x,c,t);
     yloc = C_CENTROID(2,c,t); */
  fprintf(out_file,"Hallalluja %f\n",t);
  fprintf(out_file," %f\n",tempe);
}

//-----------Start computing second and fourth terms---------------
//--------------Calculate Binary Diffusion Coefficients------------
for (i=0;i<6;i++){
  for (j=0;j<6;j++){
    sigmar[i][j] = 0.5*(sigma[i] + sigma[j]);
    epokr[i][j]  = sqrt(epok[i]*epok[j]);
    td_star[i][j] = tempe/epokr[i][j];
  }
}

for (i=0;i<6;i++){
  for (j=0;j<6;j++){
    one = A*pow(td_star[i][j],-B);
    two = C*exp(-D*td_star[i][j]);
    three = E*exp(-F*td_star[i][j]);
    omega[i][j] = one + two + three;
  }
}

for (i=0;i<6;i++){
  ooMm[i] = 1.0/Mm[i];
}
```

```
  for (i=0;i<6;i++){
    for (j=0;j<6;j++){
      numer = 0.0188*sqrt(pow(tempe,3)*(ooMm[i] + ooMm[j]));
      denom = Pabs*pow(sigmar[i][j],2)*omega[i][j];
      diffcoef[i][j] = numer/denom;
                  /* Binary diffusion coefficients calculated
                     by using the Lenard Jones parameters */
    }
  }
  //-----------------------------------------------------------------
  //--------------Calculate Thermal Diffusion Coefficient-----------
  thermcoef = -2.59*pow(10,-7)*pow(tempe,0.659);
  numsum =0.0;
  densum = 0.0;
  for (i=0;i<6;i++){
    numsum = pow(Mm[i],0.511)*X[i] + numsum;
    densum = pow(Mm[i],0.489)*X[i] + densum;
  }
  for (i=0;i<6;i++){
    middle[i] = (pow(Mm[i],0.511)*X[i]/numsum)-Y[i];
    /* Thermal diffusion coefficient*/
    tdiffcoef[i] = thermcoef * middle[i]*numsum/densum;
  }
  //-----------------------------------------------------------------
  Mmm = 0.0;
  for (i=0;i<6;i++){
    /* Calculates molecular mass at a certain cell */
    Mmm = Mm[i]*X[i] + Mmm;
  }
  //----------Create Matricies to find gradient of mole fraction -----
  for (i=0;i<5;i++){
    for (j=0;j<5;j++){
      if (i == j){
xx[i][j]=1-X[i]*(Mm[j]-Mm[5])/Mmm;
      }
      else{
xx[i][j]=-X[i]*(Mm[j]-Mm[5])/Mmm;
      }
    }
    ans1[i]=gradYx[i]*Mmm/Mm[i];
    ans2[i]=gradYy[i]*Mmm/Mm[i];
  }
  //----------------End of creating Matrix to find gradient
          //                of mole fraction
```

```
  //------Assemble the Aij and Bij matricies to calculate
          //      the mass diffusion flux
  for (i=0;i<5;i++){
    for (j=0;j<5;j++){
      if (i == j){
aii1 = (X[i]*Mmm)/(diffcoef[i][5]*Mm[5]);
aii2 = 0.0;
for (k=0;k<6;k++){
  if (k == i){
  }
  else{
    aii2 = (X[k]*Mmm)/(diffcoef[i][k]*Mm[i]) + aii2;
  }
}
DA[i][j]= -1*(aii1 + aii2);
bii1 = X[i]*Mmm/Mm[5];
bii2 = (1-X[i])*Mmm/Mm[i];
DB[i][j] = -1*(bii1 + bii2);
      }
      else{
aij1 = (1/diffcoef[i][j]) * (Mmm/Mm[j]);
aij2 = (1/diffcoef[i][5]) * (Mmm/Mm[5]);
DA[i][j] = X[i]*(aij1-aij2);
DB[i][j] = X[i]*(Mmm/Mm[j] - Mmm/Mm[5]);
      }
    }
  }
  //---------------------------------------------------------------------
  // Invert the DAij (/*1*/) and xx matrix (/*2*/) using LU decomposition//

  for (i=0;i<5;i++){
    for (j=0;j<5;j++){
      yin[i][j] = 0.0;             /*1*/ //defines an identity matrix
      DAO[i][j] = DA[i][j];        /*1*/ //saves DA[i][j]
                                         //information to check
      ide[i][j] = 0.0;                   /*2*/
      xxo[i][j] = xx[i][j];              /*2*/
    }
    yin[i][i] = 1.0;               /*1*/
    ide[i][i] = 1.0;                   /*2*/
  }

  ud = 1.0;                        /*1*/
  ud2 = 1.0;                         /*2*/
  for (i=0;i<5;i++){
```

```
    aamax = 0.0;                    /*1*/
    aamax2= 0.0;                          /*2*/
    for (j=0;j<5;j++){
      if (fabs(DA[i][j]) > aamax){ /*1*/
aamax = fabs(DA[i][j]);    /*1*/
      }                             /*1*/
      if (fabs(xx[i][j]) > aamax2){      /*2*/
aamax2= fabs(xx[i][j]);           /*2*/
      }                                   /*2*/
    }
    if (aamax < 1.0e-21){          /*1*/
      printf("Singular matrix");   /*1*/
    }                             /*1*/
    if (aamax2< 1.0e-21){               /*2*/
      printf("Singular matrix in xx");   /*2*/
    }                                   /*2*/
    vv[i] = 1.0 / aamax;           /*1*/
    vv2[i] = 1.0 / aamax2;              /*2*/
  }

  for (j=0;j<5;j++){
    if (j > 0){
      for (i=0;i<j;i++){
sum = DA[i][j];             /*1*/
sumx = xx[i][j];                  /*2*/
if (i > 0){
  for (k=0;k<i;k++){
    sum = sum - DA[i][k] * DA[k][j];  /*1*/
    sumx = sumx - xx[i][k] * xx[k][j];      /*2*/
  }
  DA[i][j] = sum;          /*1*/
  xx[i][j] = sumx;                /*2*/
}
      }
    }
    aamax = 0.0;                    /*1*/
    aamax2 = 0.0;                         /*2*/
    for (i=j;i<5;i++){
      sum = DA[i][j];               /*1*/
      sumx = xx[i][j];                    /*2*/
      if (j > 0){
for (k=0;k<(j);k++){
  sum = sum - DA[i][k] * DA[k][j];  /*1*/
  sumx = sumx - xx[i][k] * xx[k][j];      /*2*/
}
```

```
DA[i][j] = sum;            /*1*/
xx[i][k] = sumx;                  /*2*/
      }
      dum = vv[i] * fabs(sum);     /*1*/
      dum2 = vv2[i] * fabs(sumx);        /*2*/
      if (dum >= aamax){           /*1*/
imax = i;                  /*1*/
aamax = dum;               /*1*/
      }                            /*1*/
      if (dum2 >= aamax2){               /*2*/
imax2 = i;                        /*2*/
aamax2 = dum2;                    /*2*/
      }                                  /*2*/
    }
    if (j != imax){                /*1*/
      for (k=0;k<5;k++){           /*1*/
dum = DA[imax][k];         /*1*/
DA[imax][k] = DA[j][k];    /*1*/
DA[j][k] =dum;             /*1*/
      }                            /*1*/
      ud = -ud;                    /*1*/
      vv[imax] = vv[j];            /*1*/
    }                              /*1*/
    if (j != imax2){                   /*2*/
      for (k=0;k<5;k++){               /*2*/
dum2 = xx[imax2][k];              /*2*/
xx[imax2][k] = xx[j][k];          /*2*/
xx[j][k] =dum2;                   /*2*/
      }                                /*2*/
      ud2 = -ud2;                      /*2*/
      vv2[imax2] = vv2[j];             /*2*/
    }                                  /*2*/
    indx[j] = imax;                /*1*/
    indx2[j] = imax2;                  /*2*/
    if (j != 4){
      if (DA[j][j] == 0.0){        /*1*/
DA[j][j] = 1e-20;          /*1*/
      }                            /*1*/
      if (xx[j][j] == 0.0){              /*2*/
xx[j][j] = 1e-20;                 /*2*/
      }                                /*2*/
      dum = 1.0 / DA[j][j];        /*1*/
      dum2 = 1.0 / xx[j][j];
      for (i=j+1;i<5;i++){
DA[i][j] = DA[i][j] * dum; /*1*/
```

```
xx[i][j] = xx[i][j] * dum2;      /*2*/
      }
    }
  }
  if (DA[4][4] == 0.0){            /*1*/
    DA[4][4] = 1e-20;              /*1*/
  }                                /*1*/
  if (xx[4][4] == 0.0){                  /*2*/
    xx[4][4] = 1e-20;                    /*2*/
  }                                      /*2*/
  //-------------------------------------------------------------
  //------------------- Do the forward and backward substitution
  for (j=0;j<5;j++){
    for (i=0;i<5;i++){
      bb[i] = yin[j][i];           /*1*/
      bb2[i] = ide[j][i];                  /*2*/
    }
    ii=-1;                         /*1*/
    ii2=-1;                              /*2*/
    for (i=0;i<5;i++){
      ll = indx[i];                /*1*/
      ll2 = indx2[i];                      /*2*/
      sum = bb[ll];                /*1*/
      sumx = bb2[ll2];                     /*2*/
      bb[ll] = bb[i];              /*1*/
      bb2[ll2] = bb2[i];                   /*2*/
      if (ii != -1){               /*1*/
for (k=ii;k<i;k++){        /*1*/
  sum = sum - DA[i][k]*bb[k]; /*1*/
}                          /*1*/
      }                                    /*1*/
      else if (sum != 0.0){        /*1*/
ii = i;                    /*1*/
      }                                    /*1*/
      bb[i] = sum;                 /*1*/
      if (ii2 != -1){                      /*2*/
for (k=ii2;k<i;k++){             /*2*/
  sumx = sumx - xx[i][k]*bb2[k]; /*2*/
}                                /*2*/
      }                                    /*2*/
      else if (sumx != 0.0){               /*2*/
ii2 = i;                         /*2*/
      }                                    /*2*/
      bb2[i] = sumx;                       /*2*/
    }
```

```
    for (i=4;i>-1;i--){
      sum = bb[i];                  /*1*/
      sumx = bb2[i];                      /*2*/
      if (i < 4){
for (k=i+1;k<5;k++){
  sum = sum - DA[i][k] * bb[k];/*1*/
  sumx = sumx - xx[i][k] * bb2[k];/*2*/
}
      }
      bb[i] = sum / DA[i][i];      /*1*/
      bb2[i] = sumx / xx[i][i];           /*2*/
      yin2[i][j] = bb[i];          /*1*/ //inverse of DA matrix
      ide2[i][j] = bb2[i];                /*2*/  //inverse of xx matrix
    }
  }
  //----------------- End of Matrix Inversions-----------------
  //-------Assemble the Dij matrix for the diffusive mass flux vector
  for (i=0;i<5;i++){
    for (j=0;j<5;j++){
      sum = 0.0;
      sum2 = 0.0;
      for (k=0;k<5;k++){
sum = sum + yin2[i][k]*DAO[k][j];
sum2 = sum2 + yin2[i][k]*DB[k][j];
      }
      DIJ[i][j] = sum2;
    }
  }
  //-----------------------------------------------------------
  //------Calculate the Gradient of the mole fraction ----------
  for (i=0;i<5;i++){
    sumx = 0.0;
    sumy = 0.0;
    for (j=0;j<5;j++){
      sumx = ide2[i][j] * ans1[j] + sumx;
      sumy = ide2[i][j] * ans2[j] + sumy;
    }
    mgrx[i] = sumx;
    mgry[i] = sumy;
  }
  mgrx[5] = -1*(mgrx[0]+mgrx[1]+mgrx[2]+mgrx[3]+mgrx[4]);
  mgry[5] = -1*(mgry[0]+mgry[1]+mgry[2]+mgry[3]+mgry[4]);
  //-------------------------------------------------------------------

  //----------Calculate the diffusive mass flux vector  -------------
```

```
  for (i=0;i<5;i++){
    sumx = 0.0;
    sumy = 0.0;
    for (j=0;j<5;j++){
      sumx = sumx + -density*DIJ[i][j]*gradYx[j];
      sumy = sumy + -density*DIJ[i][j]*gradYy[j];
    }
    vecJx[i] = sumx - tdiffcoef[i]*gradtx/tempe;
    vecJy[i] = sumy - tdiffcoef[i]*gradty/tempe;
  }
  vecJx[5] = -1*(vecJx[0]+vecJx[1]+vecJx[2]+vecJx[3]+vecJx[4]);
  vecJy[5] = -1*(vecJy[0]+vecJy[1]+vecJy[2]+vecJy[3]+vecJy[4]);
  //---------------------------------------------------------------
  //---------Calculates the second term in entropy inequality-------

  //the Nth part I will take as oxygen.
  sum = 0.0;
  dotx = 0.0;
  doty = 0.0;
  dot = 0.0;
/* If mole fraction or mass fraction is below 10^(-9) then
     the portion of that particular species is assumed to be small
     and therefore not included in summation */
  if (yzero[1] == 't'){
      term2 = 0;
    }
  else{
      term2 = 1.0 /sdensity[1];
    }
  for (i=0;i<6;i++){
    if (i!=1){
      if (yzero[i] == 't'){
  term1 = 0;
  fprintf(tem_file, "zero mass frac. species %d, cell %d, \
                          thread %d, domain %d \n", i, c, t, d);
}
      else{
  term1 = 1.0/sdensity[i];
}
      dotx = vecJx[i]*(mgrx[i] * term1 - mgrx[1] * term2);
      doty = vecJy[i]*(mgry[i] * term1 - mgry[1] * term2);
      dot = dotx + doty;
      sum = sum + dot;
    }
  }
```

```
  secondterm = ctotal*8314*tempe*sum;
  C_UDMI(c,t,4) = secondterm;
  if (final >= 0.000001){
    fprintf(out_file,"second %f\n",secondterm);
  }
  //-------------------------------------------------------

  //---------Calculates fourth term in entropy inequality---------

  sumx = 0.0;
  sumy = 0.0;
  for (i=0;i<6;i++){
/* If mole fraction or mass fraction is below 10^(-9) then
     the portion of that particular species is assumed to be small
     and therefore not included in summation */
    if (yzero[i] == 't'){
term1 = 0;
      }
    else{
term1 = 1.0/sdensity[i];
      }
    sumx = sumx + tdiffcoef[i]*mgrx[i]*term1;
    sumy = sumy + tdiffcoef[i]*mgry[i]*term1;
  }
  epsilonx = -kl*gradtx-ctotal*8314*tempe*sumx;
  epsilony = -kl*gradty-ctotal*8314*tempe*sumy;
  xpart = epsilonx/tempe*gradtx;
  ypart = epsilony/tempe*gradty;
  fourth = xpart + ypart;
  C_UDMI(c,t,5) = fourth;
  if (final >= 0.000001){
    fprintf(out_file,"fourth %f\n",fourth);
  }
  //-----------------------------------------------------------------------
  //-------------Calculate the first term in the entropy inequality------
  for (i=0;i<2;i++){
    for (j=0;j<2;j++){
      ident[i][j] = 0;
    }
  }
  gradv[0][0]=C_DUDX(c,t);
  gradv[0][1]=C_DUDY(c,t);
  gradv[1][0]=C_DVDX(c,t);
  gradv[1][1]=C_DVDY(c,t);
  ident[0][0]=1;
```

```
  ident[1][1]=1;

  delv=gradv[0][0] + gradv[1][1];
  /* printf("delv = %f\n",delv);*/
  for (i=0;i<2;i++){
    for (j=0;j<2;j++){
      gradvt[i][j]=gradv[j][i];
      /*  printf("gradvt = %f\n",gradvt[i][j]); */
    }
  }
  for (i=0;i<2;i++){
    for (j=0;j<2;j++){
      tau[i][j]=mu*((gradv[i][j]+gradvt[i][j])-\
                        (2.0/3.0)*delv*ident[i][j]);
      /* printf("tau = %f\n",tau[i][j]); */
    }
  }
  for (i=0;i<2;i++){
    for (j=0;j<2;j++){
      tempo[i][j]=tau[i][j];
      /* printf("tempo = %g\n",tempo[i][j]); */
    }
  }
  for (i=0;i<2;i++){
    for (j=0;j<2;j++){
      sum=0.0;
      for (k=0;k<2;k++){
piece=tempo[i][k]*gradv[k][j];
/* printf("piece = %g\n", piece); */
sum=sum+piece;
/* printf("sum = %g\n", sum); */
      }
      first[i][j] = sum;
      /*  printf("first = %g\n",first[i][j]); */
    }
  }
  end=-1*(first[0][0]+first[1][1]);
  //printf("end = %f\n",end);
  C_UDMI(c,t,6) = end;
  if (final >= 0.000001){
    fprintf(out_file,"first %f\n",end);
  }
  //-----------------End of calculation of first term----------
  //-----------------Add up all terms of entropy inequality------
  entropy = C_UDMI(c,t,6) + C_UDMI(c,t,4) + \
```

```
                    C_UDMI(c,t,3) + C_UDMI(c,t,5);
  C_UDMI(c,t,7) = entropy;
  if (entropy >= 0.0){
    /* xloc = C_CENTROID(x,c,t);
       yloc = C_CENTROID(2,c,t); */
    summation = summation + entropy;
    fprintf(ent_file,"Point in Violation %f\n",entropy);
    fprintf(out_file,"official %f\n",t);
    fprintf(out_file," %f\n",entropy);
  }
}
      end_c_loop(c,t)
}
  fprintf(tem_file, "Summation = %g\n",summation);
  fclose(out_file);
  fclose(tem_file);
  fclose(ent_file);
}
```

VITA

Steven B. Chambers received his Bachelor's Degree in May 2002 in aerospace engineering from Texas A&M University. He completed his Master of Science in December 2004, also in aerospace engineering from Texas A&M University. The author can be reached at

3106 Gulf Ave

Midland, TX 79705

Publications

Chambers, S. B., Flitan, H. C., Cizmas P. G. A., Bachovchin, D., Lippert, T., and Little, D., "The Influence of *In Situ* Reheat on Turbine-Combustor Performance" *The 49th ASME International Gas Turbine Congress*, Vienna, Austria, June 2004.